University of Valladolid
Department of Computer Science
University School of Computer Science

Ph.D. Thesis

Evolutionary Framework for DNA Microarray Cluster Analysis

Presented by

José Antonio Castellanos Garzón

A dissertation submitted to the University of Valladolid for the Ph.D. Degree

Supervised by the Prof. Ph.D. Fernando Díaz Gómez

Valladolid, October 2012

.

D. Fernando Díaz Gómez, Profesor Titular de Universidad del Área de Conocimiento de Ciencias de la Computación e Inteligencia Artificial (CCIA) de la Universidad de Valladolid,

HAGO CONSTAR:

Que el trabajo de investigación que se recoge en la presente memoria, titulado **Evolutionary Framework for DNA Microarray Cluster Analysis** y presentado por **D. José Antonio Castellanos Garzón** para optar al Grado de Doctor Internacional, ha sido realizado bajo mi dirección en el Departamento de Informática de la Universidad de Valladolid.

Valladolid, 10 de octubre del 2012.

Dr. Fernando Díaz Gómez
Profesor Titular de Universidad
University of Valladolid

A mis padres (Enma Dora y José)
y a mi futura niña Eva Dora.

Contents

List of Tables

List of Figures

Summary

THIS research proposes an *evolutionary framework* where a method of hierarchical clustering represented by an evolutionary model, a set of cluster validation measures and a cluster visualization tool have been fused to create a suitable environment of knowledge discovery from DNA microarray data. On one hand, the clustering evolutionary model of our framework is a novel alternative that attempts to solve some of the problems faced by the existing clustering methods. On the other hand, our alternative of cluster visualization given by a tool couples new properties and visual components, allowing us to validate and analyze clustering results. It also allows a visual checking environment of the cluster validation measures. This way, the fusion of the clustering evolutionary model with the cluster visual model becomes our framework a novel application of *data mining* compared to the conventional methods of *machine learning*.

Motivation, Hypothesis and Goals

In order to reach our proposal, we have focused our efforts on the combination of areas such as *evolutionary computation*, data mining and *visual analytics* to build the framework on the domain of gene expression data. Each of these areas provides techniques that play a major role for the analysis and resolution of the current challenges in *Bioinformatics*.

The study of gene expression data from DNA microarrays is of great interest for Bioinformatics (and functional genomics), because it allows us to analyze expression levels in hundreds of thousands of genes in a living organism sample. This feature makes gene expression analysis a fundamental tool of research for human health. It provides

identification of new genes that are key in the genesis and development of diseases. However, the exploration of these large data sets is an important but difficult problem. Information visualization techniques can help to cope with this problem. Visual data exploration has high potential and many applications in data mining use information visualization technology for an improved data analysis.

Consequently, the part of the clustering evolutionary model of our framework has been motivated by the fact that the biological data representation in form of a dendrogram is one of the ways of knowledge discovery. Due to this, if the best dendrogram on a data set is found, then the following challenges could be reached:

- The optimum data representation is given by that dendrogram.
- This way, the optimum data partition is given from one of the levels (clusterings) of such a dendrogram.
- And the optimum cluster number is also obtained from one of the clusterings of such a dendrogram.

All these challenges describe forms of knowledge discovery from the problem domain, for which, the existing clustering methods are insufficient to capture knowledge from the complex processes at cellular level.

The need of defining an extensible visualization tool of cluster analysis (extensible, in the sense that new clustering methods as well as new statistical measures can be added without making meaningful changes in the tool) able to combine existing visualizations with the novel ideas of our approach, which is addressed to capitalize on added value gained from the interaction between the approaches and thus maximize the benefits to the user; it has motivated us to introduce the visual analytics part as a tool in order to reach our final goal, the evolutionary framework. Based then on the foregoing, our research hypothesis states that:

The problem of finding the best dendrogram on a data set (denoted as PFBD), that is, the problem of finding an optimum dendrogram from a data set and according to an objective function measuring the dendrogram quality is an NP-complete problem. In consequence, we propose evolutionary techniques to face it on the domain of DNA microarray data.

According to the computational complexity shown by PFBD, the second part of our hypothesis arises, stating that *based on evolutionary techniques we can define a novel method of hierarchical clustering able to improve the existing hierarchical clustering techniques, facing the problem of convergence towards local optimums.* This method has been given from an evolutionary computational model as a theoretical-practical result of this research.

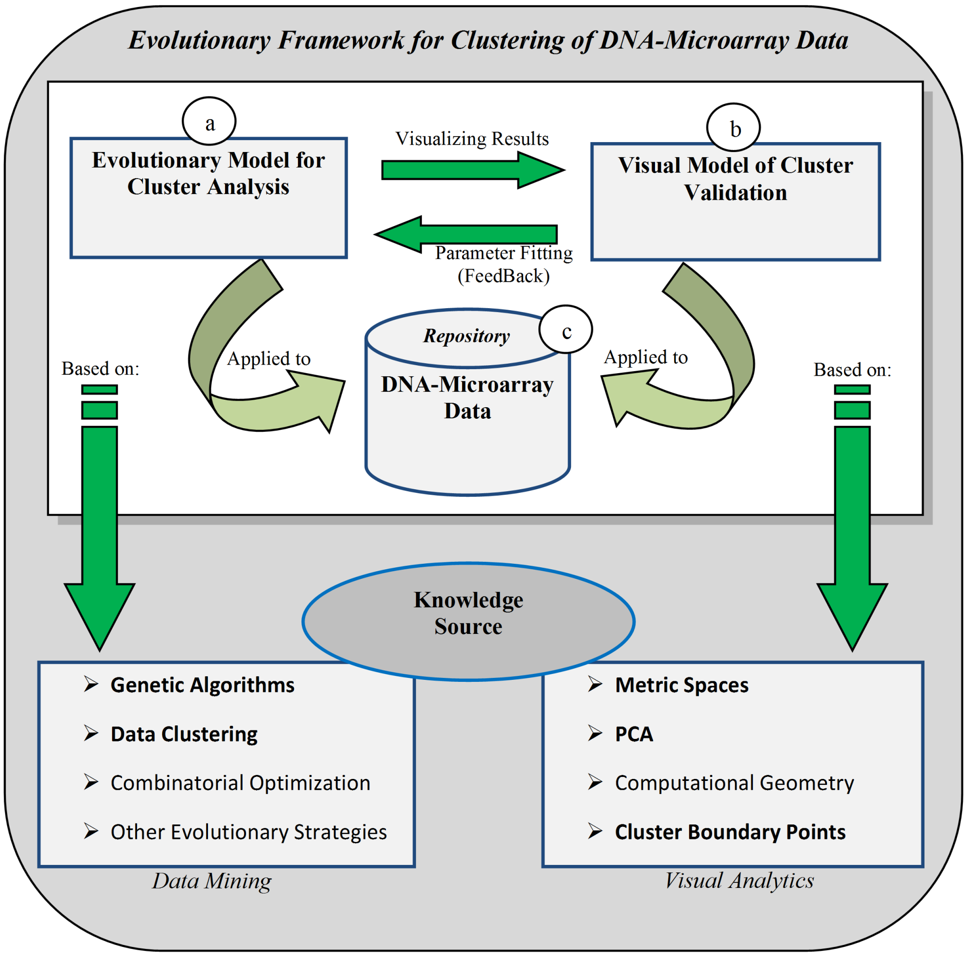


Figure 1: Evolutionary Framework for cluster analysis from DNA microarray data.

Complementing the second part of our hypothesis and given the difficulty of evaluating (and comparing) the results provided by the used clustering methods from the final user point of view in the context of DNA microarray data. We have introduced a visual framework based on visual analytics, where clustering methods (including our method), cluster validity measures and visual components of cluster exploration are integrated to improve the global process of cluster analysis. Figure 1 shows a general scheme of our evolutionary framework, where Figure 1-a represents the evolutionary model of clustering that setting its parameters, we obtain a specific clustering method on the data repository in Figure 1-c. This way, parameters and results are validated using the visual model in Figure 1-b. Moreover, both models in this figure are based on a well-known knowledge source. Note that our evolutionary framework has been extended to analyze any other hierarchical clustering method different from our evolutionary method in Figure 1-a. To develop our global framework as a join of an evolutionary model of clustering and a visual tool for cluster analysis, the following goals have been proposed:

- Characterizing the whole search space to obtain better understanding of the space and the relationships between the dendrograms. This allows us to introduce heuristics to improve the search process.
- Introducing an clustering evolutionary method able to find better solutions than other methods on the domain of gene expression data analysis.
- Building a visualization tool that represents the visual part of our evolutionary framework in order to verify and validate the results achieved by the clustering methods.
- Comparing our results with the ones of other methods on gene expression data, aimed at knowledge discovery from both, the data and the used methods.

Evolutionary Model and Visual Framework

As a part of the evolutionary framework, this research has proposed a novel hierarchical clustering method using genetic algorithms (GAs) for the analysis of gene expression data. This method is based on the mathematical proof of several results, showing its effectiveness with regard to other clustering methods. GAs applied to cluster analysis have disclosed good results on biological data and many studies have been carried out in this sense, although most of them are focused on partitional clustering methods. Even though there are a few studies that attempt to use GAs for building hierarchical clustering, they do not include constraints that allow us to reduce the complexity of the problem. Therefore, these studies become intractable problems for large data sets. On the other hand, the deterministic hierarchical clustering methods generally face the problem of convergence towards local optimums due to their greedy strategy. The method introduced here is an alternative to solve some of the problems existing methods face.

According to the previously mentioned, the goal of this approach has been the search of clustering hierarchies of high quality on DNA microarray data. For reaching that aim, other contributions have been given, such as: a specific method (called EMHC, Evolutionary Model for Hierarchical Cluster[1]) from a clustering evolutionary model is obtained by prefixing the parameters of such a model [6, 7]. This method provides a novel fitness function to evaluate dendrograms based on the cluster definition. In this context several strategies (constraints) are introduced to reduce the complexity of the search space. That is, reduction of the level number of a dendrogram based on non-valuable information, reduction of the fitness function runtime by introducing two fundamental lemmas, and the partition of the search space in neighborhoods to state differences between local and global optimum.

We have also introduced two novel genetic operators (mutation and crossover) performing an agglomerative and divisive strategy to build the child dendrograms. In order to carry out in-depth search to improve the solutions given by our method, several evolutionary strategies of local search have been built. Another important result found from our experiments is that the use a genetic algorithm is not enough to deal with the problem of finding an optimum dendrogram in the search space. Hence, we have introduced several constraints and heuristics to make our intractable problem, feasible in an approximate way. All these contributions have allowed our method to find better solutions compared to the other methods.

Finally, note that this evolutionary approach has gone further than just defining an algorithm as in other researches. The idea consists of creating an evolutionary

[1]Manual and the software-package published under *R-Project* licence at `http://cran.r-project.org/web/packages/clustergas`

model where a set of parameters can be pre-fitted based on some criterium, so that we can obtain a concrete clustering method able to adapt to the analyzed problem, but varying those parameters we will possibly achieve a different method. Such an approach is possible thanks to evolutionary computation.

The Visual Framework

To complement our evolutionary model of clustering we have developed a visual analytics framework to be used in cluster analysis from gene expression data. In addition, this visual framework has presented a novel method of finding cluster boundaries based on the theory of metric spaces. Our visual approach links a set of visualizations able to interact with parallel coordinates, cluster boundary points on a 3D scatter plot (using dimensionality reduction) and DNA microarray visualizations. Thus, it is also a visual alternative with respect to the cluster validity measures currently used. Besides that, the method of computing cluster boundary is also used to estimate the shape that a cluster has on a 3D-space, and represent reference partitions (on a 3D-space) coming from the problem domain.

This visual framework has introduced data exploration for aggregating, summarizing and visualizing information generated during interactive cluster analysis from DNA microarray data [8–11]. As a result of the visual part of our global framework (the evolutionary framework), we have developed a prototype tool called *3D-VisualCluster* (or 3D-VC) [12], which is able to explore dendrograms, clusterings and clusters interactively with different views [13,14]. This prototype uses *principal component analysis* (PCA) to reduce data dimensionality to $\mathbb{R}^3$, so that a first approximation of data distribution can be analyzed on a 3D scatter plot. Furthermore, parallel coordinate visualization [15] and DNA microarray data views (heat map) have also been presented by using a color scale corresponding to gene expression levels.

The new visualizations of 3D-VC have been combined with other existing visualizations of our approach through linked and interactive views. This way, the visual analytics process is reached. On the other hand, as our prototype has been linked to the *R language* [16], the results reached by clustering methods implemented on R can be displayed by the 3D-VC tool. Since R is a programming language used in Bioinformatics, almost all clustering methods are developed on R through software packages.

Results and Conclusions

This section presents the results reached in the evaluation studies of the defined evolutionary framework through the 3D-VisualCluster tool (3D-VC) and the proposed evolutionary method (EMHC). To do that, we have analyzed the 3D-VC reliability of providing new knowledge through its visualizations from the clustering results and the cluster validity measures applied to gene expression data. Additionally, we have studied the behavior of EMHC on three public data sets of gene expression data and compared the results with other methods according to cluster validity measures. The results of EMHC have also been analyzed with visualizations given by framework 3D-VC.

The analysis of the results have been made by detailing the scenario where 3D-VC and EMHC have been tested, that is, stating data sets, hierarchical clustering methods and cluster validation measures to use. After that, firstly, the evaluation and analysis of framework 3D-VC have been made on a practical case study. Secondly, EMHC has been evaluated from a wide number of statistical tests and visualizations given by 3D-VC.

On 3D-VC, three types of validation for the results of three hierarchical clustering methods have been completed. That is, we have validated the results using indices of internal cluster validation, using the dendrogram and microarray view, and finally by visual comparison with a reference partition of the used data set. In this context, the most meaningful clusters according to the reference partition have been identified. Moreover, a set of tasks has been defined from the views of the tool in order to introduce a methodology to follow by the user in visual cluster analysis and validation of clustering results. All this have shown the importance of the 3D-VC tool in cluster analysis of DNA microarray data[2].

Furthermore, the evaluation of EMHC has been made on its genetic operators and afterwards, the goodness of the individuals generated by the evolutionary process of the method has been evaluated. After that, three sections have been given, one for each data set, dedicated to compare the results of EMHC with regard to the considered methods and using the 3D-VC tool. We can say that EMHC performs better than the others, on cluster separation measures and on measures that combine homogeneity and separation (such as, silhouette width), but it does not have the same performance on

[2]Additional material on the tool performance at `http://www.analiticavisual.com/jcastellanos/3DVisualCluster/3D-VisualCluster`

homogeneity. It should be taken into account that most of the hierarchical clustering methods focus on cluster internal quality (homogeneity) and not on cluster external quality (adding separation), that is, global quality. For this reason, the other methods work better on homogeneity. However, EMHC looks for two or three indicators, homogeneity, separation and number of genes in each cluster, so that we can check two or three objectives on the clusters at a time. What is more, the visualizations of 3D-VC have also shown that EMHC performs well and EMHC can even improve its solutions from solutions given by other methods. All these results have been possible because we have carried out a study of the complexity of the given problem (proving that PFBD is NP-complete), which allowed us to extract knowledge of the search space. This way, we have focused our efforts on finding good approximated solutions to the problem instead of trying to find a polynomial algorithm that solves the problem in a non-approximate way (that is, an algorithm running in polynomial time with respect to its input).

As a final conclusion of our proposal, we state that, the fusion of both approaches to create the evolutionary framework has proved to be of great help in the understanding of the data and provided knowledge on the used clustering methods. Moreover, we have provided new visualization components which can also be used to validate the existing ones. The framework has proven that our evolutionary method performs well and that can find better solutions than the others, this is shown not only through validity measures but also, with result visualizations. This way, the visual part of our framework is able to display clustering results and validate both, cluster validity measures and more importantly, a visualization component with another. Consequently, the results have shown that our framework is a powerful tool for cluster analysis from DNA microarray data, not only for the introduced evolutionary model, but also for any other hierarchical clustering method that we want to add to the process of cluster analysis.

Resumen

EN esta investigación se propone un *framework evolutivo* donde se fusionan un método de clustering jerárquico basado en un modelo evolutivo, un conjunto de medidas de validación de agrupamientos (*clusters*) de datos y una herramienta de visualización de *clusterings*. El objetivo es crear un marco apropiado para la extracción de conocimiento a partir de datos provenientes de *DNA-microarrays*. Por una parte, el modelo evolutivo de *clustering* de nuestro *framework* es una alternativa novedosa que intenta resolver algunos de los problemas presentes en los métodos de *clustering* existentes. Por otra parte, nuestra alternativa de visualización de *clusterings*, materializada en una herramienta, incorpora nuevas propiedades y nuevos componentes de visualización, lo cual permite validar y analizar los resultados de la tarea de *clustering*. El *framework* propuesto permite a su vez, disponer de un medio para comprobar la adecuación de las medidas de validación de agrupamientos. De este modo, la integración del modelo evolutivo de *clustering* con el modelo visual de *clustering*, convierta a nuestro *framework* evolutivo en una aplicación novedosa de *minería de datos* frente a los métodos convencionales de *aprendizaje automático*.

Motivación, Hipótesis y Objetivos

Para llevar a cabo nuestra propuesta, nos hemos basado en disciplinas como la *computación evolutiva*, la minería de datos y la *analítica visual* para construir el *framework* sobre el dominio de datos de expresión génica. Cada una de estas áreas proporciona técnicas que juegan un papel importante en el análisis y resolución de los retos actuales en *Bioinformática*.

El estudio de datos de expresión génica a partir de *DNA-microarrays* es de gran interés en Bioinformática (y en genómica funcional) ya que éste permite analizar niveles de expresión de cientos de miles de genes en una muestra de un organismo vivo. Esta característica hace que el análisis de expresión génica, sea una herramienta fundamental en la investigación de la salud humana, proporcionando así, un medio para identificar nuevos genes, claves en la génesis y desarrollo de enfermedades. Sin embargo, si bien es cierto que la exploración de estos grandes conjuntos de datos es una tarea importante, también es cierto que es un problema difícil. Las técnicas de visualización de la información pueden ayudar a resolver el problema. La exploración visual de datos, tiene un gran potencial y muchas aplicaciones de minería de datos emplean técnicas de visualización para mejorar el rendimiento del análisis de *cluster* (análisis de agrupamientos).

Consecuentemente, la inclusión del modelo evolutivo de *clustering* (jerárquico) en nuestro *framework* se debe a que la representación de datos biológicos en forma de un dendograma, es una de las formas de extracción de conocimiento. Debido a esto, si se encuentra el mejor dendograma sobre un conjunto de datos, entonces sería posible alcanzar la siguientes metas:

- La representación óptima de los datos está dada por ese dendograma.
- De esta manera, la partición óptima de los datos estaría en uno de los niveles (*clusterings*) de tal dendograma.
- Y el número óptimo de agrupamientos (*clusters*) estaría también en uno de los niveles de ese dendograma.

Cualquiera de estas metas representan una forma de extraer conocimiento "práctico" a partir del dominio del problema, para el cual, los métodos de *clustering* actuales son aún insuficiente en la extracción de conocimiento de los complejos procesos ocurridos a nivel celular.

La necesidad imperante por definir una herramienta de visualización extensible (extensible, en el sentido de que se puedan incorporar nuevos métodos de *clustering* y nuevas medidas estadísticas, sin realizar cambios significativos en la herramienta) nos ha motivado a introducir la parte de analítica visual como un componente más y así, alcanzar nuestro objetivo final, el *framework* evolutivo. Esta herramienta combina las visualizaciones existentes con las nuevas ideas de nuestro enfoque, y añadirá valor al proceso de análisis de *cluster*, mediante la interacción de las técnicas empleadas, maximizando, por tanto, los beneficios para el usuario. Basado entonces en todo lo anterior, nuestra hipótesis de investigación plantea que:

El problema de encontrar el mejor dendograma sobre un conjunto de datos (que denominaremos PFBD), es decir, el problema de encontrar un dendograma óptimo sobre un conjunto de datos y respecto a una función objetivo que mide la calidad de los dendogramas, es un problema NP-completo. En consecuencia, proponemos técnicas evolutivas para enfrentar dicho problema en el dominio de datos de DNA-microarrays.

Partiendo de la complejidad computacional que implica el problema PFBD, surge la segunda parte de nuestra hipótesis de trabajo, la cual establece que *basándonos en técnicas evolutivas, podemos definir un nuevo método de clustering jerárquico, capaz de mejorar las técnicas existentes de clustering jerárquico, las cuales presentan, en general, el problema de convergencia hacia óptimos locales.* Tal método ha sido desarrollado a partir de un modelo computacional evolutivo, siendo un resultado práctico-teórico de esta investigación.

Complementando la segunda parte de nuestra hipótesis y debido a la dificultad (a nivel operativo) para evaluar y comparar los resultados de los métodos de *clustering* empleados por el usuario final, en el contexto de datos de *DNA-microarrays*; se ha introducido un *framework* visual basado en la eficacia de la analítica visual, donde métodos de *clustering* (incluyendo el nuestro), medidas de validación de *clusters* y componentes visuales de exploración de *clusters*, son integrados, con el objetivo de mejorar el proceso global del análisis de *cluster*. En la Figura 2 se muestra un esquema general de nuestra propuesta, el *framework* evolutivo, donde la Figura 2-a representa el modelo evolutivo de *clustering* que mediante la configuración de sus parámetros, permite definir un método específico de *clustering*, que se aplicará a un *dataset* almacenado en el repositorio de datos (Figura 2-c). De esta manera, parámetros y resultados pueden validarse utilizando el modelo visual de la Figura 2-b. Además, tanto el modelo evolutivo como el modelo visual están asentados sobre un *corpus* de conocimiento (teórico) bien conocido. Por otra parte, se ha de hacer notar que el *framework* evolutivo propuesto es extensible, de modo que no sólo se analice nuestro método, sino que también, permita analizar cualquier método *clustering*, en general. Entonces, para

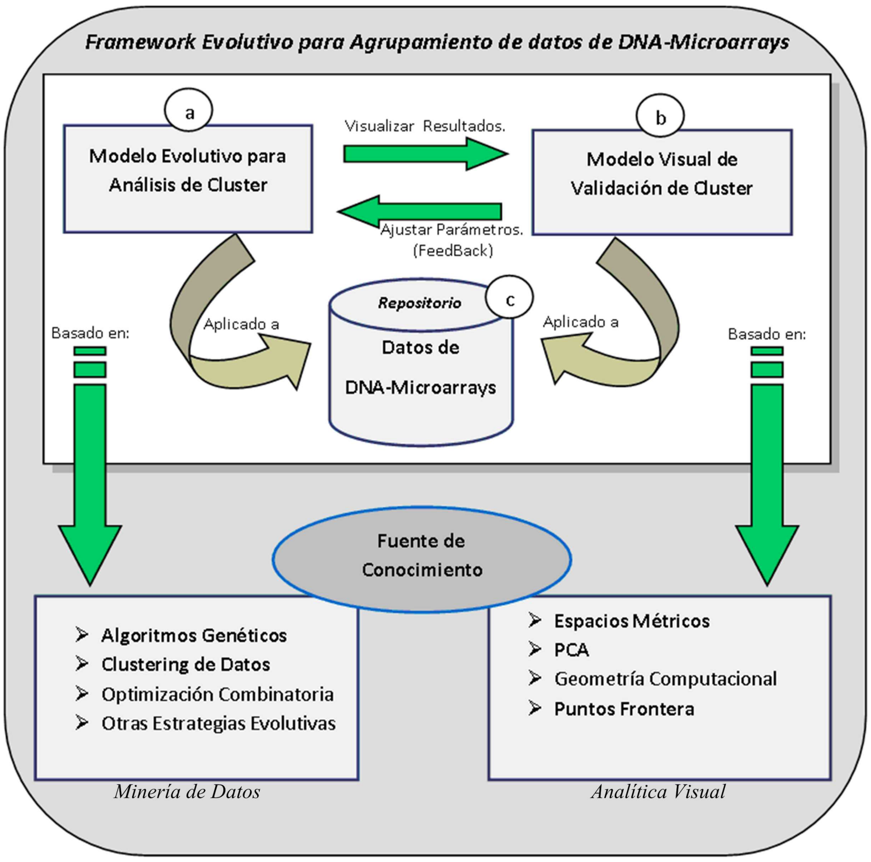


Figure 2: *Framework* Evolutivo para el análisis de *cluster* sobre datos de *DNA-microarrays*.

desarrollar el *framework* evolutivo como unión de un modelo evolutivo de *clustering* y una herramienta visual de análisis de *cluster*, se propusieron los siguientes objetivos en el marco del presente trabajo de investigación:

- Caracterizar el espacio de búsqueda, para así, comprender mejor el espacio y las relaciones existentes entre los dendogramas. Esto permitirá introducir heurísticas que mejoren el proceso de búsqueda.

- Introducir un método evolutivo de *clustering* capaz de encontrar mejores soluciones que los otros métodos, en el dominio del análisis de datos de expresión génica.

- Construir una herramienta de visualización que represente la parte visual de nuestro *framework* evolutivo, con el objetivo de verificar y validar los resultados alcanzados por los métodos de *clustering*.

- Comparar nuestros resultados con los resultados de otros métodos, a partir de datos de expresión génica, todo ello dirigido a la extracción de conocimiento, tanto a partir de los datos como de los métodos empleados.

Modelo Evolutivo y Framework Visual

En esta investigación se ha propuesto, como parte del *framework* evolutivo, un nuevo método de *clustering* jerárquico basado en algoritmos genéticos (AGs) y orientado al análisis de datos de expresión génica. Este método está sustentado sobre la demostración formal de varios resultados teóricos, mostrando de esta manera, su efectividad respecto a otros métodos de *clustering*. En en el ámbito de la Bioinformática, la aplicación de los AGs al análisis de agrupamientos, ha revelado buenos resultados sobre datos biológicos y consecuentemente, se han realizado múltiples estudios en este sentido, aunque la mayoría de ellos se centran en los métodos de *clustering* particionales. Aún cuando existen unos pocos estudios que intentan aplicar los AGs a la construcción de *clustering* jerárquico, éstos no incluyen restricciones que permitan reducir la complejidad del problema. En consecuencia, tales estudios se convierten en problemas intratables para grandes conjuntos de datos. Por otra parte, los métodos deterministas de *clustering* jerárquico, presentan el problema de convergencia hacia óptimos locales, debido al carácter voraz de sus estrategias algorítmicas. De aquí que, nuestro método sea una alternativa para resolver algunos de los problemas existentes en los métodos de *clustering* actuales.

De acuerdo con lo anterior, el objetivo de este enfoque ha sido encontrar jerarquías de *clusterings* de alta calidad sobre datos de *DNA-microarrays*. Para alcanzar dicho objetivo se han realizado varias contribuciones en el marco de esta investigación, tales como: la definición de un método específico (llamado EMHC, modelo evolutivo para *clustering* jerárquico[3]) construido a partir de prefijar los parámetros de un modelo

[3]Manual y el paquete-software publicado bajo licencia de *R-Project* en `http://cran.r-project.`

evolutivo de *clustering* [6, 7]. Junto con este método, se define una función de aptitud que evalúa los dendogramas basándose en la definición de agrupamiento (*cluster*). En este contexto, se han introducido varias estrategias (y restricciones) con el objetivo de reducir la complejidad del espacio de búsqueda. A saber, la reducción del número de niveles de los dendogramas basándonos en la información no valiosa de éstos, la reducción del tiempo de ejecución de la función de aptitud a partir de la introducción de dos lemas fundamentales, y la partición del espacio de búsqueda en vecindades con el fin de diferenciar un óptimo local de un óptimo global.

Por otra lado, también hemos introducido dos nuevas versiones de operadores genéticos de mutación y cruce, las cuales, a partir de una estrategia aglomerativa y divisiva, permiten construir los dendogramas hijos. Con el objetivo de realizar una búsqueda en profundidad, y mejorar así las soluciones encontradas por nuestro método, se han implementado también varias estrategias evolutivas de búsqueda local. Otro resultado importante, obtenido a partir de los experimentos realizados, fue que, un algoritmo genético no es suficiente para tratar con el problema de la búsqueda de un dendograma óptimo. En consecuencia, hemos introducido varias restricciones y heurísticas para transformar este problema intratable, *a priori*, en un problema tratable. Todas estas contribuciones han permitido que nuestro método pueda encontrar mejores soluciones que los otros métodos.

Finalmente, ha de hacerse notar que el enfoque propuesto ha tratado de ir más allá de la mera definición de un algoritmo de *clustering*. La idea ha consistido en crear un modelo evolutivo parametrizable, de modo que el establecimiento de un conjunto de parámetros, en base a diferentes criterios, permita obtener un método de *clustering* concreto, capaz de adaptarse al problema tratado. De este modo, variando dichos parámetros se obtendrá, posiblemente, un método de *clustering* diferente. Tal enfoque es posible gracias a la computación evolutiva.

El Framework Visual

Complementando nuestro modelo evolutivo de *clustering*, se ha desarrollado un componente de analítica visual para su uso en la tarea del análisis de *cluster* sobre datos de expresión génica. Adicionalmente, en este modelo visual se ha definido un nuevo método de construcción de la frontera de un *cluster*, basado en la teoría de los espacios métricos.

`org/web/packages/clustergas`

Asimismo, este enfoque enlaza un conjunto de visualizaciones, interactuando entre sí, a través de componentes visuales como: coordenadas paralelas, puntos fronteras sobre un espacio 3D (a través de la reducción de la dimensionalidad) y visualizaciones de *DNA-microarrays*. Por tanto, este enfoque visual es también una alternativa respecto a las medidas de validación de *cluster*, actualmente usadas. Por otra parte, el método que calcula la frontera de un *cluster* se utiliza también para estimar la forma de un *cluster* y para representar particiones de referencias (en un espacio 3D) provenientes del dominio del problema.

De acuerdo con lo anterior, nuestro enfoque visual se ha introducido con el objetivo de agregar, resumir y visualizar, la información generada durante el proceso de análisis de *cluster* interactivo, a partir de datos de *DNA-microarrays* [8–11]. Entonces, como un producto final de nuestro *framework* global (el *framework* evolutivo), se ha desarrollado una herramienta prototipo, llamada *3D-VisualCluster* (3D-VC) [12], que permite explorar interactivamente, dendogramas, *clusterings* y *clusters* sobre diferentes vistas [13, 14]. En este prototipo, se utiliza el *análisis de componentes principales* (PCA) para reducir la dimensionalidad de los datos a $\mathbb{R}^3$, para de este modo, poder realizar un análisis preliminar de los datos mediante una vista de puntos 3D. Además, a todo esto se une las visualizaciones de coordenadas paralelas [15] y las vistas de *DNA-microarrays* (*heat map*), las cuales se generaron a partir de una escala de colores que se corresponde a los niveles de expresión génica.

Las nuevas visualizaciones de la herramienta 3D-VC se combinan con visualizaciones existentes, lo que permite a esta herramienta materializar el proceso de analítica visual, mediante la visualización de vistas interactivas y enlazadas. Por otra parte, y dado que nuestra herramienta permite interactuar con la API del *lenguaje R* [16], los resultados devueltos por cualquier método de *clustering* implementado en R, pueden ser leídos y analizados visualmente, desde la herramienta. Además, como R es un lenguaje de programación empleado en Bioinformática, casi todos los métodos de *clustering* están implementados en R a través de los correspondientes paquetes de software, desarrollados por la comunidad Bioinformática.

Resultados y Conclusiones

En este apartado comentamos los resultados obtenidos, a partir de los estudios de evaluación del *framework* evolutivo, a través de la herramienta 3D-VisualCluster (3D-VC) y del método evolutivo propuesto EMHC. Para llegar a ésto, por una parte, se ha analizado la utilidad de 3D-VC en la extracción de conocimiento a partir de los resultados *clustering* y de las medidas validación de *clusters* en los experimentos realizados sobre datos de expresión génica. Por otra parte, se estudió también, el comportamiento del método EMHC sobre tres conjuntos de datos públicos de expresión génica y comparamos los resultados con otros métodos conocidos, utilizando medidas convencionales de validación de *clusters*. En la evaluación de los resultados del método EMHC, nos basamos también, en el *framework* visual 3D-VC.

Así pues, para llevar a cabo el análisis de los resultados, primero, se detalló el escenario donde se evaluarían 3D-VC y EMHC, es decir, se detallaron los conjuntos de datos, los métodos de *clustering* y las medidas de validación de *clusters* a utilizar. Como siguiente paso, se realizó la evaluación y el análisis del *framework* 3D-VC sobre un caso de estudio práctico. Finalmente, se evaluó el método EMHC, empleando un amplio número de pruebas estadísticas y visualizaciones de la herramienta 3D-VC.

Sobre la herramienta 3D-VC, se plantearon tres escenarios de validación a partir de los resultados obtenidos por tres métodos de *clustering*. A saber, validación utilizando índices internos de validez de *clusters*, utilizando vistas de dendogramas y *microarrays* (*heat maps*), y finalmente, comparando visualmente, los resultados con una partición de referencia del conjunto de datos utilizado por los tres métodos de *clustering*. En este contexto, también se identificaron los agrupamientos (*clusters*) más significativos dentro de los resultados obtenidos, desde el punto de vista de la partición de referencia. Además, se definió un conjunto de tareas sobre las visualizaciones de la herramienta, que establecen un metodología a seguir por el usuario en el análisis *cluster* visual y en la validación de los resultados de *clustering*. Todo esto demostró la validez de la herramienta 3D-VC en el análisis de *clusters* de datos provenientes de *DNA-microarrays*[4].

Respecto al método EMHC, primero se evaluó la validez de los operadores genéticos (de cruce y mutación) definidos, para luego, evaluar la bondad de los individuos generados

[4]La herramienta y material adicional sobre su funcionalidad en `http://www.analiticavisual.com/jcastellanos/3DVisualCluster/3D-VisualCluster`

a lo largo del proceso evolutivo desarrollado por el método. A partir de aquí, se plantearon otros tres experimentos, uno para cada conjunto de datos considerado, para comparar los resultados del método EMHC frente a los resultados obtenidos por otros métodos *clustering* bien conocidos, utilizando asimismo, la herramienta 3D-VC para este fin. De estos experimentos, se obtuvo que EMHC funcionó mejor que los demás métodos en lo que respecta a medidas de separación de *clusters* y medidas que combinan, separación y homogeneidad (como por ejemplo, el ancho de silueta), no siendo así para medidas exclusivas de homogeneidad. En este sentido, se debe considerar que la mayoría de los métodos de clustering centran su atención en mejorar la calidad interna de los *clusters* (homogeneidad o distancia intra-*cluster*) y no sobre la calidad externa (considerando la separación o distancia inter-*cluster*), que puede entenderse como una medida más global de la calidad del *clustering*. Debido a esta razón, los demás métodos de *clustering* funcionan mejor sobre medidas de homogeneidad. Sin embargo, EMHC optimiza dos o tres objetivos a la vez, como son, homogeneidad, separación y número de genes en cada *cluster*. Además, las visualizaciones mediante la herramienta 3D-VC demostraron también el buen desempeño de EMHC en relación con la mejora de soluciones previas, considerando para ello, la inclusión como población inicial (de individuos) en proceso evolutivo de EMHC, las soluciones obtenidas mediante otros métodos. Todos estos resultados fueron posibles, debido al estudio realizado sobre la complejidad del problema tratado (demostrando que PFBD es NP-completo), lo que permitió obtener más conocimiento del espacio de búsqueda, y así, enfocar todos los esfuerzos en encontrar buenas soluciones aproximadas al problema en vez de intentar de encontrar un algoritmo eficiente que resuelva el problema de forma no aproximada (o sea, un algoritmo que se ejecute en tiempo polinomial respecto a su entrada).

Como conclusion final de nuestro trabajo de investigación, tenemos que la fusión de ambos enfoques para crear el *framework* evolutivo, resultó de gran ayuda en la comprensión de los datos y proporcionó también, conocimiento de los métodos de *clustering* empleados en el análisis de *cluster*. Además, nuestro *framework* incluyó nuevos componentes de visualización, los cuales pueden validar las visualizaciones ya existentes. Así pues, a partir del *framework* evolutivo, se ha demostrado el buen funcionamiento de nuestro método evolutivo y que éste puede encontrar mejores soluciones que otros métodos conocidos de *clustering* jerárquico. Esto se demostró no sólo a través de medidas de validez de *clusters*, sino que también, con visualizaciones de los resultados.

De esta forma, la parte visual de nuestro *framework* evolutivo es capaz de visualizar resultados de *clustering*, útiles para evaluar las medidas de validación de *clusters* y lo que es más importante, validar un componente de visualización con otro. Consecuentemente, los resultados han mostrado que nuestro *framework* evolutivo es una potente herramienta del análisis de *clusters* a partir de datos de *DNA-microarrays* y que no sólo es útil para nuestro modelo evolutivo de *clustering*, sino que en general, lo es para todos los métodos de *clustering* que se quieran añadir al proceso del análisis de *clusters*.

Agradecimientos

Sin lugar a dudas, el trabajo presentado en esta memoria, no hubiese sido posible sin la orientación, ayuda y experiencia de mi director de tesis, el Prof. Dr. Fernando Díaz Gómez. Al cual, quiero agradecer todo su esfuerzo, dedicación y valor humano, empleados en el desarrollo de esta investigación para que la actual memoria fuese posible.

Quiero dar las gracias también a Carlos A. García (investigador en formación de la Universidad de Salamanca), compañero de doctorado, por su colaboración desinteresada y su gran ayuda en el desarrollo de la herramienta *3D-VisualCluster*, la cual juega un papel fundamental en esta investigación.

Agradecer al Prof. Dr. Luis Alonso Romero, Universidad de Salamanca, cuyos oportunos comentarios sobre formalizar ciertos conceptos en los inicios de esta investigación, dio un vuelco positivo a la investigación hacia su avance.

Agradecer también, al Prof. Dr. Miguel A. Borges Trenard de la Universidad de Oriente (director de mi tesis de master en matemática), Cuba, del cual he obtenido formación en el campo de las matemáticas y me he nutrido hasta los días actuales, de su experiencia tanto en la investigación como en la vida.

De mi formación como doctor, agradezco las oportunas discusiones, comentarios y en su caso, ayuda, de mis compañeros de doctorado, en especial, Antonio Gonzáles, Saddys Segrera, el Prof. Dr. Cristian Pinzón, Diego A. Gómez y el Dr. Juan F. García, sobre diferentes temas de la investigación y el doctorado.

Además, las diferentes etapas por la que ha pasado esta investigación, han sido también posibles debido a la financiación total o parcial de las siguientes instituciones: La Agencia Española de Cooperación Internacional (MAEC-AECI, 2004-2007); El Ministerio de Ciencia e Innovación (MICINN) en el proyecto ALIADO (Alzheimer Intelligent Ambient Domotic system, 2008-2010) FIT-350300-2007-84 y en El Plan E (a partir del 2011) del Gobierno de España y la Unión Europea, ERDF (TIN2009-14057-C03-02).

Por otra parte y a nivel afectivo, agradecerle a María, mi esposa y compañera, su amor, paciencia, comprensión y ayuda a lo largo de las diferentes etapas del desarrollo de mi doctorado. De hecho, destacar su aporte en la revisión de la gramática de muchos de los textos en inglés, relacionados a esta investigación. Al mismo nivel, agradecerle a mi querida hermana Laertys por su ayuda, suministrando apoyo, preocupación y su espíritu luchador, inculcado por nuestros padres, en todo mi recorrido académico, incluso, desde que estuve haciendo la licenciatura. Finalmente, dar las gracias por su apoyo a mi querida familia (cubana, española y griega), en especial a mi familia en Cuba (el resto de mis hermanos), que sin dudas ha llevado la peor parte, viviendo mi ausencia todos estos años.

Sería muy difícil llenar con estás líneas toda la gratitud que debo a todos los que han puesto su grano de arena para alcanzar la presente memoria. Pero al menos, con estos agradecimientos, quiero demostrar que este trabajo no ha sido fruto único de mi esfuerzo, sino que en este largo camino, han contribuido muchas personas, y a todos ellos, gracias.

Chapter 1

Introduction

IN this research work, we propose an *evolutionary framework* where a method of hierarchical clustering represented by an evolutionary model, a set of cluster validation measures and a cluster visualization tool are integrated to create a suitable environment of knowledge discovery from DNA microarray data. On one hand, the clustering evolutionary model of our framework is a novel alternative that attempts to solve some of the problems faced by the existing clustering methods. On the other hand, our alternative (for cluster visualization) given by a tool couples new properties and visual components, allowing us to validate and analyze clustering results. It also creates a visual checking environment for cluster validity measures. This way, the fusion of the clustering evolutionary model with the cluster visual module becomes our framework a novel application of *data mining* compared to the conventional methods of *machine learning*.

According to the all above, we have focused our efforts on the combination of areas such as *evolutionary computation*, data mining and *visual analytics* to build the framework on the domain of gene expression data. Each of these areas provides techniques that play a major role for the analysis and resolution of the current challenges in *Bioinformatics*.

Within evolutionary computation, *evolutionary algorithms* represent a powerful tool to solve complex optimization problems where traditional methods can hardly find an optimum solution. Hence, they carry out an efficient, adaptive and robust search providing optimization processes that are usually applied to very large, complex and multidimensional search spaces.

The application of evolutionary algorithms to data mining is still of great importance in classification problems. Particularly, we can highlight, the use of evolutionary strategies to *unsupervised classification* in the knowledge discovery process. *Cluster analysis* as an unsupervised classification task is part of machine learning as well as a way of finding out structures on the data of a given problem. Classification of similar objects into groups is an important task of human activity, being part of daily learning process.

In recent years, there has been a growing interest in applying evolutionary, cluster analysis and visualization techniques to Bioinformatics, which is one of the most controversial areas of research at present, since it deals with the development and/or application of methods and algorithms to turn biological data into knowledge of biological systems, often requiring further experimentation from initial data [38].

Consequently, the study of gene expression data from DNA microarrays is of great interest for Bioinformatics (and functional genomics), because it allows us to simultaneously analyze expression levels from hundreds of thousands of genes in a living organism sample. This feature makes gene expression analysis a fundamental tool of research for human health. It provides identification of new genes that are key factors in the genesis and development of diseases. However, the exploration of these large data sets is an important yet difficult problem. Information visualization techniques can help to face this problem. Visual data exploration has high potential and many applications in data mining use information visualization technology for an improved data analysis.

In conclusion, the present research has been motivated by all issues previously raised, providing a starting point to state the research hypothesis. This chapter additionally introduces the global motivation, evolution and structure of this research work.

1.1 Motivation

Advances in Bioinformatics have resulted in a great output of biological data, which has been made available in different public databases. Biologists often wish to analyze these databases in order to understand the relationship between their items. Traditional approaches are oriented to discover evolutionary relationships between genes or proteins by analyzing their sequence and structure similarities [13].

Future work involves the tight integration of visualization techniques with traditional techniques from disciplines as *statistics, machine learning, operations research* and *simulation*. Integration of visualization techniques within these more established methods would combine fast automatic data mining algorithms with the intuitive power of the human mind, improving the quality and speed of the visual data mining process [14].

Moreover, it is known that gene expression data generated by microarray experiments provide tremendous potential for advances in molecular biology and functional genomics. According to that, there is a great number of clustering algorithms applied to the analysis of DNA microarray data. However, most of them only provide a crisp set of clusters and may not be flexible to different user requirements on cluster granularity for different subsets of the data [137]. Evolutionary methods are more flexible with respect to the above problems, due to both, their non-deterministic nature that allows them to find more than one solution to the same problem, and their ability to consider user requirements.

Consequently, the part of the clustering evolutionary model of our framework is motivated by the fact that the biological data representation in the form of a dendrogram is one of the ways of knowledge discovery. Due to this, if the best dendrogram on a data set is found, then the following challenges could be reached:

- The optimum data representation is given by that dendrogram.
- This way, the optimum data partition is given from one of the levels (clusterings) of such a dendrogram.
- Consequently, the optimum cluster number is also obtained from one of the clusterings of such a dendrogram.

All these challenges describe forms of knowledge discovery from the problem domain, for which, the existing clustering methods are insufficient to capture knowledge from the complex processes at cellular level.

On the other hand, it has arose the need of defining an extensible visualization tool for cluster analysis (extensible, in the sense that new clustering methods and statistical measures can be added without making meaningful changes in the tool) able to combine existing visualizations with the novel ideas of our approach, addressed to capitalize on added value gained from the interaction between the components and thus maximize the benefits to the user. All these issues motivated us to introduce the visual analytics part as a tool addressed to reach our final goal, the evolutionary framework.

Therefore, based on the success of the evolutionary techniques applied to cluster analysis from DNA microarray data, and the benefits obtained by their integration with information visualization techniques; this research proposes an evolutionary hierarchical clustering method that combined with a cluster visualization tool, builds a framework able to improve the results given from other hierarchical clustering methods and provides new visualization components to explore and validate clustering results.

1.2 Research Hypothesis

Starting from the premises given in the above section, we can state our research hypothesis as follows:

The problem of finding the best dendrogram on a data set (denoted as PFBD), that is, the problem of finding an optimum dendrogram from a data set, according to an objective function measuring the dendrogram quality is an NP-complete problem. In consequence, we propose evolutionary techniques to face it on the domain of DNA microarray data.

Our hypothesis presents two well defined parts which help to develop the stages of this research. The first part states the conjecture that the problem of searching the best dendrogram from a given data set is NP-complete. This means that not only it is NP-hard, it also belongs to the class of the hardest problems in NP. Hence, until now, no polynomial algorithm has been found for the problems in this NP-complete class. They include many well-known and important properties from the theoretical and practical point of view, for which there is an intense worldwide research effort into understanding them in recent years. Due to the importance of classifying our problem (PFBD), we have dedicated Chapter 5 to characterize the dendrogram search space, measure the complexity of the problem and finally, give proof of its NP-completeness.

According to the computational complexity shown by PFBD, arises the second part of our hypothesis, which states that based on evolutionary techniques, we can define a novel method of hierarchical clustering able to improve the existing hierarchical clustering techniques, which face the problem of convergence towards local optimums. This method is given by an evolutionary computational model in Chapter 6 as a theoretical-practical result.

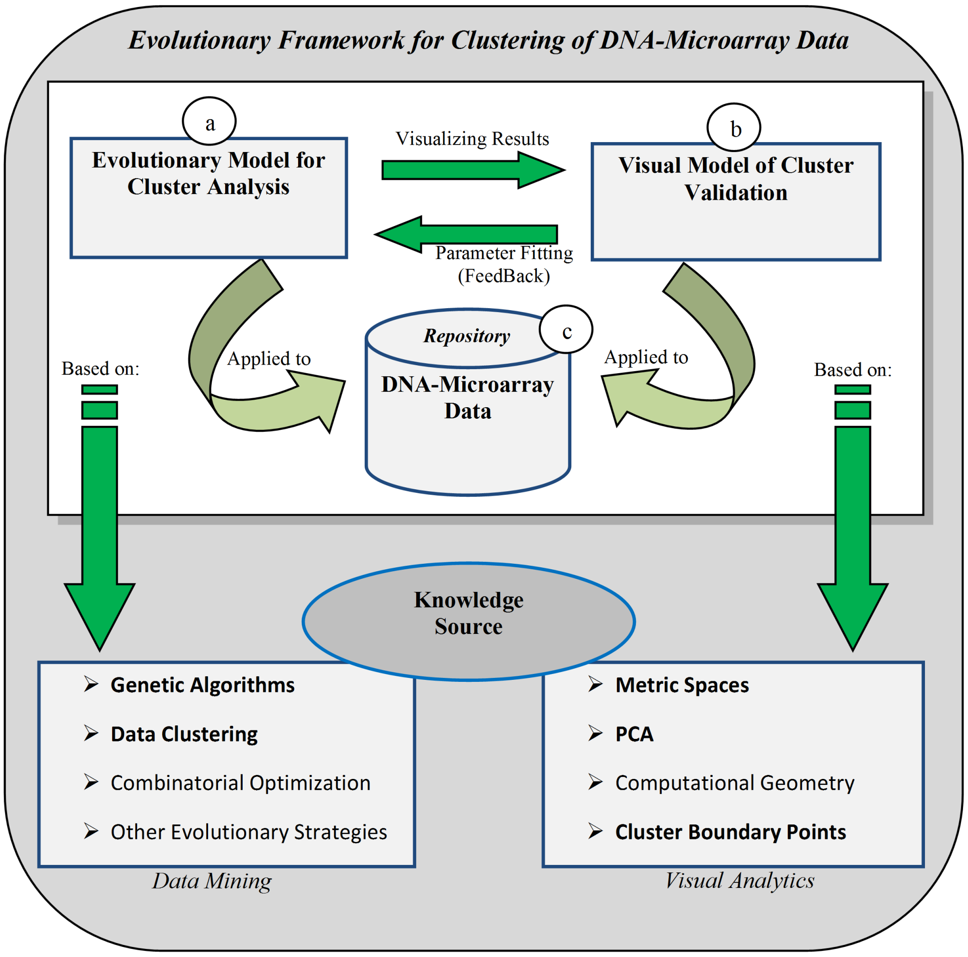


Figure 1.1: Evolutionary Framework for cluster analysis from DNA microarray data.

Complementing the second part of our hypothesis and given the difficulty of evaluating (and comparing) the results provided by the current clustering methods from the end user point of view in the context of DNA microarray data. We have introduced a visual framework based on visual analytics techniques, where clustering methods (including our method), cluster validity measures and visual components of cluster exploration are integrated to improve the global process of cluster analysis. Figure 1.1 shows a general scheme of such an evolutionary framework, where Figure 1.1- a represents the evolutionary model of clustering which by setting its parameters, gives us a specific clustering method for a data set stored in our repository in Figure 1.1- c. This way, parameters and results are validated using the visual model in Figure 1.1- b. Moreover, both models in this figure are based on a well-known source of knowledge. Note that this framework can be extended to analyze any other hierarchical clustering method different from our evolutionary method.

The definition and theoretical results of this visual framework are given in Chapter 7. According to this, the global reliability of the visual framework and the evaluation of the introduced evolutionary method (both form the evolutionary framework) from DNA microarray data are empirically proven in Chapter 8. Based then on all fails previously explained, the following goals will be met:

- Characterizing the whole search space to obtain better understanding of the space and the relationships between the dendrograms. This allows us to introduce heuristics to improve the search process.

- Introducing a clustering evolutionary method able to find better solutions than other methods on the domain of gene expression data analysis.

- Building a visualization tool that represents the visual part of our evolutionary framework in order to verify and validate the results achieved by any clustering method.

- Comparing our results with the ones of other methods from gene expression data, aimed at knowledge discovery from both, the data and used methods.

1.3 Research Evolution

This research had its starting point on the results and experiments concluded from the implementation of a genetic algorithm for data clustering given in [46], which was applied to DNA microarray data. Although in this case, good results were achieved on the studied data set, this algorithm presented several drawbacks requiring improvement in such a way that it can perform on larger data sets.

Therefore, the above work has successively been improved and validated through a series of contributions presented in conferences and published in journals (see Appendix A). Each of these works meant an improvement of our global research. The first step was to study the nature of the problem from the theoretical point of view, for which we introduced constraints and new properties to improve the performance on DNA microarray data. Another important consequence of the study above was to change the previous approach (build a genetic algorithm for hierarchical clustering) by the one that studies *the problem of finding the best dendrogram on a data set.* This last study allowed us to characterize (and understand) the search space and so, define a new evolutionary model of cluster analysis in such a way that by setting its parameters, we can obtain a specific method.

In particular, the inclusion of the neighborhood concept, the reduction of the size of the dendrograms based on non-valuable information (noise) and the reduction of the computation complexity of the used objective function, allowed us to introduce new heuristics that improved the search process and the effectiveness of the genetic operators of our current evolutionary method, yielding high quality dendrograms from DNA microarray data. The implementation of the evolutionary model has been carried out on R Project [16] and is publicly available at `http://cran.r-project.org/web/packages/clustergas`.

From the need of visually exploring the results of our approach, evaluating and comparing them with other methods through internal and external measures of cluster validation, arises the currently proposed framework that provides a great number of linked visual components. This integrates our part of data mining with the part of visual analytics to achieve a powerful tool of cluster analysis from DNA microarray data. Finally, the introduced framework has been implemented in Java and Java 3D, yielding the first prototype which is available at `http://www.analiticavisual.com/jcastellanos/3DVisualCluster/3D-VisualCluster`.

1.4 Outline

The present work has been divided into nine chapters plus several appendices, for which the first chapter brings, in general aspects, the problems to solve, the starting hypothesis with the goals and the evolution stages of this research, concluding with the organizational structure in this PhD Thesis.

Chapter 2 deals with microarray technology, development stages and its challenges. This chapter gives an introduction to DNA microarray data analysis. Furthermore, we give a literature review of the available main sources of biological knowledge and their use for the interpretation of results from DNA microarray data.

Chapter 3 outlines domain, metric and representation of the data before dealing with clustering methods. It defines the concepts of clustering, hierarchical clustering, clustering method, cluster validity and the importance of visualizing the results. We finally give a literature review describing the existing clustering and visualization techniques used for the interpretation of results from DNA microarray data.

Chapter 4 deals with concepts, principles and problems when we model a given problem with evolutionary algorithms. It describes the main goal of an optimization process according to the use of genetic algorithms as well as tips to improve their performance. We also give a literature review of evolutionary algorithms applied to hierarchical clustering analysis from DNA microarray data.

Note that at the end of the previous theoretical chapters, that is, Chapters 2, 3 and 4, we have made a literature review about the addressed subject from DNA microarray data. This is very important since the reader can contrast the contributions and drawbacks of the existing techniques with regard to our proposal.

Chapter 5 states the first contribution of this work, namely, the fundamental theorem of this research based on our hypothesis. As a consequence of this theorem, we provide the characterization, understanding and complexity of the dendrogram search space, to finally give formal proof of the fundamental theorem. All these results will be of great importance for the following chapters.

As a result of Chapter 5 and the second part of our hypothesis, Chapter 6 defines an evolutionary computational model for hierarchical clustering analysis, which will be applied to DNA microarray data.

Chapter 7 introduces the theoretical basis of the visual analytics part of the evolutionary framework, which is based on metric spaces. Additionally, this chapter defines two algorithms, one to compute the boundary points of a cluster and the another one, to reconstruct 3D cluster surfaces. Both algorithms will be used later in visual cluster analysis. At the end of the chapter, we give an overview of the visualizations that this visual model provides and details of its implementation.

Chapter 8 shows the results of this research on DNA microarray data (results of the evolutionary framework). It presents the data sets of DNA microarrays and the hierarchical clustering methods used to generate the results. On one hand, we present the visual components coupled to our evolutionary framework, which are integrated as a tool of visual cluster analysis. The reliability of this tool is shown from a practical case. On the other hand, we show the results and discussion of our evolutionary clustering method, supporting us on such a visualization tool.

The conclusions of this research are presented in Chapter 9. We comment the results achieved from the experiments and their repercussion in the analysis of DNA microarray data. We also explain the reached goals and the future lines of this work.

In addition, we also provide a set of appendices most of which them are related to Chapters 7. That is, Appendix A outlines a list of the scientific publications related to this work. Appendix B describes the technique of data dimensionality reduction used by the framework defined in Chapter 7. Appendix C states definitions and properties of the space metric theory useful for building our framework introduced in Chapter 7. Appendix D formally states the theoretical results reached by the visual framework defined in Chapter 7. Finally, the bibliography used in this research is given at the end of this document.

Chapter 2

Microarray Technology and its Challenges

DATA mining and functional genomics have recently gained attention since several complete genome sequences as well as the human genome were published. One of the most advanced and challenging way of studying molecular events is the monitoring of gene expression patterns from DNA microarrays. Microarrays can view as a type of device (a chip) in which, a large number of diverse entities, such as peptides, oligonucleotides, biological molecules, cells, tissues, etc., are located on its surface in an ordered and accurate way. Once these entities are attached on the surface of the chip, they can simultaneously be evaluated in a single assay.

DNA microarrays allow to attach hundreds of thousands of DNA fragments (*deoxyribonucleic acid*) with determined sequences, on specific and defined positions of their surfaces. This kind of microarray is primarily used in differential gene expression studies (that is, the comparison of the genes that express a tissue affected by some kind of disease with respect to healthy tissue). This way, expression profiles of samples can be obtained with a prognostic capacity.

From a simple experimental device given by DNA microarray technology, researchers can monitor interactions of thousands of gene transcriptions in an organism. This technology is particularly useful in the evaluation of gene expression patters during important biological processes and across collections of related samples. So, DNA microarrays are able to observe at the same moment and in response to the same stimulus, the gene expression levels of many genes under different samples.

This chapter presents a description of DNA microarray technology as well as a general vision of the related knowledge sources and their use. To do this, we first introduce DNA microarray technology explaining the manufacture process, technical characteristics and the different types of microarrays. Thereafter, we explicate the information processing and the analysis techniques most commonly applied on DNA microarrays. Through DNA microarrays are obtained the experimental data on which the framework proposed in this work is applied.

2.1 Measuring mRNA Levels

In contrast to the traditional approaches of genomic research, which focus on the local examination and collections of single genes, microarray experiments can monitor expression levels for tens of thousands of genes in parallel. The two types of DNA microarrays most commonly used are: *cDNA microarrays* (complementary DNA [168]) and *oligonucleotide arrays* (abbreviated oligo chip [212]). Both microarray types involve three basic processes in their implementations [238]:

- *Chip confection*: a microarray is formed by a solid basis of an area less than 20 cm^2 (made of chemically coated glass, nylon membrane, or silicon (Figure 2.1-a), which acts as a succession of contiguous grids with genetic material from known sequences. Every sequence on the grids represents independent experiment tests in presence and abundance of specific sequences of bases from a polynucleotide chain. About tens of thousands of DNA molecules (*probes*) are attached (stuck) to fixed grids, relating each grid cell to a DNA sequence (called *gene*, see Figure 2.1-b). Affymetrix (pioneer enterprise in the development of microarray technology) builds these probes one layer at a time, using the same type of manufacturing technology that is used to build computer semiconductors. The molecules are built one layer at a time, one stacked on top of another, like bricks. Multiple probes are synthesized in parallel.

- *Target preparation, labeling, and hybridization*: typically, two mRNA samples, a test sample and a control sample (more than two samples for oligo chips) are reverse transcribed into cDNA (*targets*). Afterwards, these samples are labeled using either fluorescent dyes or radioactive isotopics, and then hybridized with the complementary sequences on the surface of the chip (see Figure 2.2). In a multi-step process, researchers extract RNA from the sample and make millions of copies. Copying the RNA allows it to be more easily detected on the array. The entire prepared RNA sample is washed over the array for 14 to 16 hours allowing that the hybridization occurs. The number of molecules involved in this wash is staggering. There are millions of copies of each DNA probe in every square on the chip. All of the RNA strands from expressed genes are swimming around, looking for their perfect molecular complement on the array. Note that if many target polynucleotides hybridize to complementary cDNA probe strands at one spot on the array then the fluorescent signals emitted and detected at that spot will have greater intensity.

- *Scanning process*: Chips are scanned to read the signal intensity that is emitted from the labeled and hybridized targets (Figure 2.3). Through this process we can measure how much of RNA strands have stuck to the DNA probe on the array. If a gene is highly expressed, many RNA molecules will stick to the probe, and the probe location will shine brightly when the laser hits it. If a gene was expressed at a lower level, less RNA will stick to the probe, and by comparison, that probe location will be much dimmer when it is hit with the laser.

Generally, the cDNA microarray and oligo chip experiments measure the expression level of each DNA sequence by the ratio of signal intensity (Figure 2.3-b) between the test sample and the control sample, therefore, data sets resulting from both methods share the same biological semantics. So, we focus on the resulting data set, regardless of the method used to obtain it. All values collected in a data set as a result of applying those methods are called *gene expression levels*, [35, 38, 100].

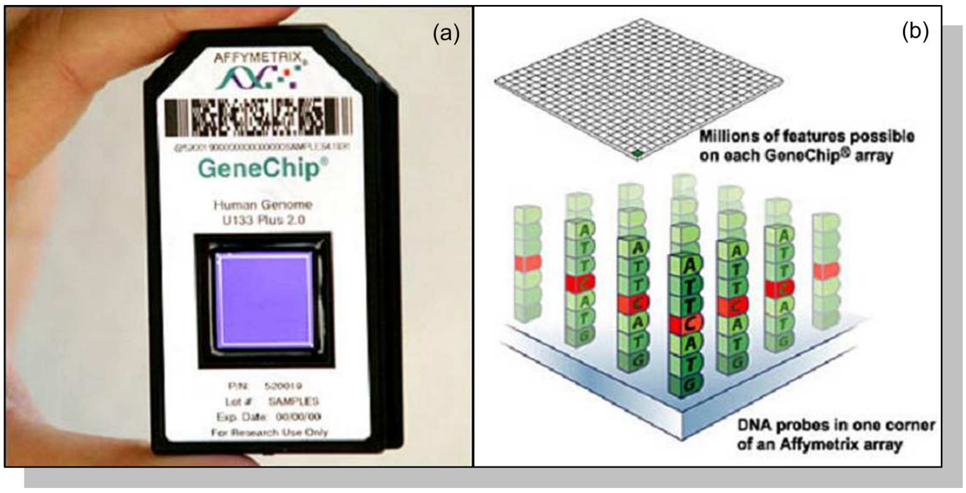

(a)
AFFYMETRIX
GeneChip
Human Genome U133 Plus 2.0
P/N 520019
Lot # SAMPLES
Exp. Date 00/00/00
For Research Use Only
(b)
Millions of features possible on each GeneChip® array
DNA probes in one corner of an Affymetrix array

Figure 2.1: a) displays a GeneChip array of Affimetrix and b) displays the grid surface of it, where DNA probes of known sequences are attached.

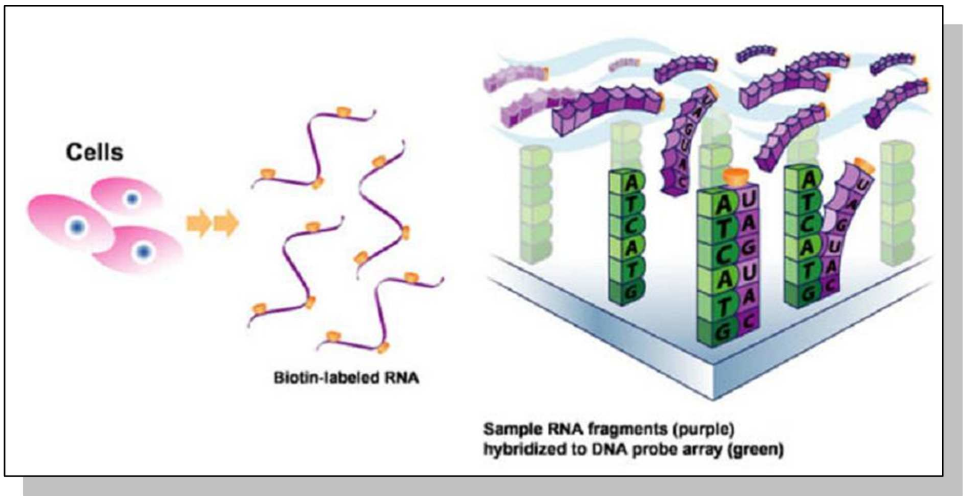

Cells
Biotin-labeled RNA
Sample RNA fragments (purple) hybridized to DNA probe array (green)

Figure 2.2: Summary of the hybridization process from DNA complementary sequences to the DNA probe array of the chip.

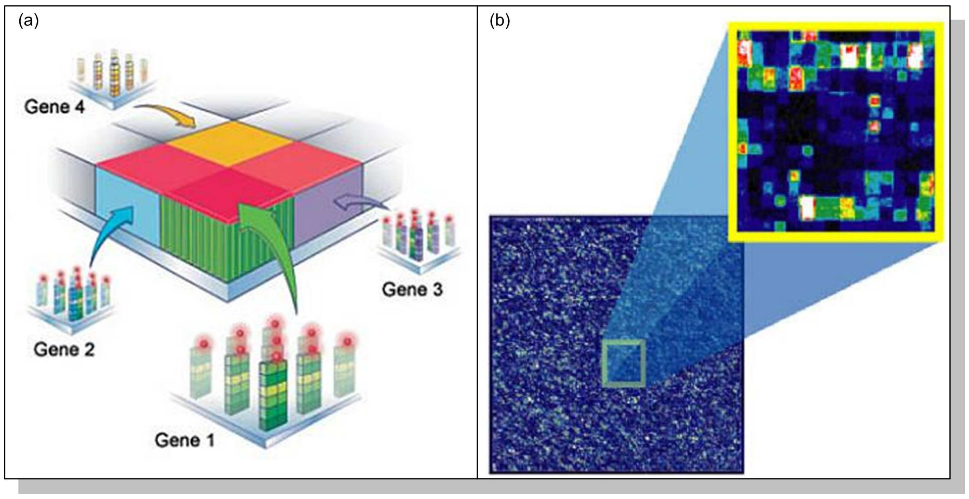


Figure 2.3: a) displays the intensity levels (through colors) captured by grids of some genes after the hybridization. b) displays the resulting image of the whole microarray from the scanning process.

2.2 Gene Expression Data Preprocessing

One experiment from a DNA microarray typically, evaluates a large number of DNA sequences (genes, cDNA clones, or expressed sequence tags) under multiplies conditions. These conditions may be a time series during a biological process (e.g., the yeast cell cycle) or a collection of different tissue samples (e.g., normal versus cancerous tissues). In this research, cluster analysis is made without considering distinctions between the types of DNA sequences, which are called *genes*. Similarly, the conditions uniformly refer to all kinds of experimental conditions, called *samples* or simply *conditions*.

A gene expression data set from a microarray experiment is represented through a matrix of real values $M = \{w_{ij} / \ \ 1 \leq i \leq n, 1 \leq j \leq m\}$, called *gene expression matrix*, where n is the number of genes evaluated for m samples. Rows $G = \{\vec{g}_1, \vec{g}_2, \ldots, \vec{g}_n\}$ form the expression patterns of genes, columns $S = \{\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_m\}$ represent the expression profiles of samples and every cell w_{ij} is the measured expression level of gene i in sample j, as shown in Figure 2.4.

The gene expression matrix extracted from the scanning process, usually, presents noise, missing values and systematic variations arising from the experimental procedure.

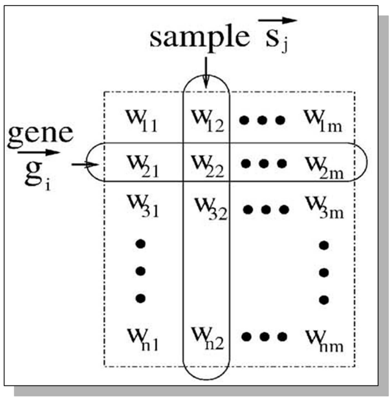

sample $\vec{s}_j$
gene $\vec{g}_i$
w_{11} w_{12} ••• w_{1m}
w_{21} w_{22} ••• w_{2m}
w_{31} w_{32} ••• w_{3m}
w_{n1} w_{n2} ••• w_{nm}

Figure 2.4: Gene expression matrix of a microarray.

Thus, it is indispensable before any cluster analysis can be performed, to carry out a data preprocessing [8, 35, 100, 122, 134, 241].

2.3 Clustering for Gene Expression Data Analysis

Clustering is the process of partitioning a data set into disjoint classes, which are called *clusters*. Thus, objects within a cluster are more similar than objects located in different clusters. Clustering carries out a unsupervised task, which means that it does not rely on predefined classes and training examples while classifies the data objects. The issue about clustering is addressed in more details in the next chapter.

Clustering techniques have shown a great potential on the understanding of the gene function, gene regulation, cellular processes, and subtypes of cells. Genes with similar expression patterns (*coexpressed genes*) can be clustered together with similar cellular functions. Coexpressed genes in a same cluster may be involved in the same cellular processes, and a strong correlation of expression patterns between those genes indicates coregulation. The inference of regulation through gene expression data clustering reinforces the hypothesis considerations on the mechanism of transcriptional regulatory network [66]. On the other hand, sample-based clustering; which is based on profile similarity, can disclose subcell types that are hard of finding from traditional approaches [20,107].

2.4 Clustering Categories of Gene Expression Data

Currently, one microarray experiment, approximately includes between 10^3 and 10^6 genes. However, the number of samples involved in a microarray experiment is generally less than 100. One of the main characteristics in gene expression data is its capacity of carrying out clusterings of both genes and samples. Coexpressed genes can group in clusters based on their gene expression patterns [33,86]. In a *gene-based clustering*, the genes are considered as the objects to make clusters, while the samples are treated as the features of these objects.

In contrast to gene-based clustering, the samples can also be partitioned into homogenous clusters. Then, each cluster may be related to some particular macroscopic phenotype, such as a clinical syndrome or some cancer type [107]. Hence, *sample-based clustering* treats the samples as the objects and the genes as the features.

The distinction between gene-based and sample-based clustering is focused on different characteristics of clustering tasks for gene expression data. Both types of clusterings look for exclusive and exhaustive partitions of objects that share the same feature space (genes or samples). Nevertheless, criteriums from molecular biology state that *only a small subset of genes participate in any cellular process of interest and that a cellular process takes place only in a subset of the samples* [137]. This belief leads to the emergence of *subspace clustering* (or *biclustering*), which captures clusters from a subset of genes under a subset of samples.

For the subspace clustering algorithms, genes and samples are treated symmetrically, so that either genes or samples can be regarded as objects or features indistinctly. Moreover, clusters generated through such algorithms may have different feature spaces. Based on the above, we have that each one of the three types of clusterings faces very different challenges and therefore, it may adopt very different computational strategies for each case.

2.4.1 Gene-based Clustering

This subsection discusses the clustering problem of genes through their expression levels. The aim of gene-based clustering is to group together coexpressed genes, indicating cofunction and coregulation. So that the challenges of this approach, which are still open problems [137] are focused on:

1. Clustering analysis is typically the first task in data mining and knowledge discovery. The purpose of gene expression data clustering is disclosing the natural structure of data and so, to obtain an initial understanding from data distribution. Thereby, a good clustering algorithm should depend as little as possible on prior knowledge, which is usually not available before cluster analysis.

2. Since gene expression data coming from complex procedures of microarray experiments, often contains a huge amount of noise; a clustering algorithm for gene expression data should be able to extract valuable information from the high noise level in the data.

3. Empirical studies [135,136] have shown that gene expression data are often *highly connected*, and clusters may be highly intersected with each other or even embedded one in another. Hence, algorithms for gene-based clustering should be capable of suitably handling this situation.

4. A clustering algorithm apart of partitioning gene expression data, should provide some graphic representation of the cluster structure. In that way, an algorithm would be more favored by the biologists. Because, users of microarray data could not only be concerned to the gene clusters, but also be concerned to the existing relationship between the clusters, and the relationship between the genes within the same cluster. Namely, which clusters are either closer or more remote to each other and on the other side, which gene can be considered as the representative of a cluster and which genes are at the boundary area of a cluster.

In order to solve some of the previous points, different approaches have been used, such as: K-means [184], Self-Organizing Map (SOM) [152], Hierarchical clustering [10,84], Graph-Theoretical (CLICK, CAST) [217,251], model-based clustering [95,103, 187,255], density-based hierarchical clustering (DHC) [135]. However, those algorithms were designed for the general purpose of clustering, and may not be effective to address the particular challenges for gene-based clustering [137].

On the other hand, different clustering algorithms are based on different criteriums and/or different assumptions regarding data distribution. The performance of each algorithm may vary greatly with different data sets and there is no a “winner” at all.

2.4.2 Sample-based Clustering

The aim of sample-based clustering is to find the phenotype structures or substructures of the samples. There are usually several particular macroscopic phenotype of samples related to some diseases or drug effects, such as diseased samples, normal samples, or drug treated samples. Studies from molecular biology ([107]) have shown that *phenotypes of samples can be discriminated through only a small subset of genes whose expression levels strongly correlate with the class distinction.* These genes are called *informative genes* and the remaining genes in the gene expression matrix are considered as irrelevant to the classification of samples of interest. So, the latter is assumed as noise of the data set.

Although the conventional clustering methods, such as K-means, self-organizing maps (SOM), hierarchical clustering (HC), can be directly applied to cluster samples using all the genes as features, the number of informative genes versus that of irrelevant genes is usually smaller than 1 : 10, which may seriously degrade the quality and reliability of clustering results [233, 251]. Hence, novel methods should be applied to identify informative genes and reduce gene dimensionality for clustering samples to detect their phenotypes.

The existing methods of informative gene selection are classified in *supervised analysis* (clustering based on supervised informative gene selection) and *unsupervised analysis* (unsupervised clustering and informative gene selection). The supervised approach assumes that the samples are labeled by a phenotype. Using this information, a classifier can be built by only containing the informative genes. Then, the samples can be clustered according to their phenotypes, and labels can be predicted for the future coming samples from the expression profiles.

Unsupervised sample-based approach assumes that does not exist information on the categories of sample phenotypes. This makes unsupervised approach more complex than the previous approach since no training sample set is available to guide the selection of informative genes. For this reason, many of statistic methods and other supervised methods can not be applied without the phenotypes of samples known in advance.

2.4.3 Subspace Clustering

At this point we have only seen approaches of *global clustering*, that is, the feature space (is determined globally) is shared by all resulting clusters, which are exclusive and exhaustive. However, it is known from molecular biology that many activation patterns are common to a group of genes only under specific experimental conditions. This reasoning on cellular processes, leads us to expect subsets of genes to be coregulated and coexpressed only under certain experimental conditions, but to behave almost independently under other conditions. It follows that a single gene may participate in multiple pathways that may or may not be coactive under all conditions, so that a gene can participate in multiple clusters or in none at all [175].

In that way, discovering such local expression patterns may be the key to uncovering many genetic pathways that are not apparent otherwise. Hence, it is highly desirable to go beyond the clustering paradigm and to develop approaches capable of discovering local patterns in microarray data [31].

Subspace clustering was proposed (by Agrawal et al. [18]) in an effort to find subsets of objects such that the objects appear as a cluster in a subspace formed by a subset of the features. The subsets of features of several biclusters (subspace clusters) can be different. On the other hand, two biclusters may share some common objects and features, and some objects may not belong to any bicluster.

We can say that in a bicluster, genes and samples are treated symmetrically such that either genes or samples can be regarded as objects or features. This means that biclustering derives a local model while clustering produces a global model. Therefore, biclustering approaches are the key technique to use when one or more of the following situations applies [175]:

1. Only a small set of the genes participates in a cellular process of interest;
2. An interesting cellular process is active only in a subset of the conditions;
3. A single gene may participate in multiple pathways that may or may not be coactive under all conditions.

Hence, a biclustering identifies groups of genes and conditions, obeying the following restrictions:

1. A cluster of genes should be defined with respect to only a subset of the conditions;
2. A cluster of conditions should be defined with respect to only a subset of the genes;
3. The clusters should not be exclusive and/or exhaustive: a gene/condition should be able to belong to more than one cluster or to no cluster at all and be grouped using a subset of conditions/genes.

To conclude, different methods have been used according to the above requirements, such as those in [41, 44, 52, 57, 102, 117, 151, 165, 166, 213, 214, 232, 235, 252, 253], which are aimed at finding biclusters for four major classes:

1. Biclusters with constant values;
2. Biclusters with constant values on rows or columns;
3. Biclusters with coherent values, and
4. Biclusters with coherent evolutions.

2.5 Microarray Data Sources and their Uses

This section addresses the information schemes and the most important data bases from DNA microarrays. On the other hand, we also explain the main uses currently given to microarray technology.

The growing volume of scientific publications related to DNA microarrays along with the need of making public their data sets in order to verify and extract conclusions from the data, has generated public repositories to store, index and share data from carried out experiments. In that sense, some public repositories for array-based gene expression data [100] are given by: International sequence data bases include the GenBank®, [National Center for Biotechnology Information (NCBI), National Institutes of Health, Bethesda, Maryland], the DNA DataBank of Japan (DDBJ), at the Center for Information Biology in Mishima, Japan; and the European Molecular Biology Laboratory (EMBL), suported by el European Bioinformatics Institute in Cambridgeshire, UK. Through an agreement known as the International Nucleotide Sequence Database Collaboration, the three organizations conduct daily exchanges of data through the Internet. GenBank® contains nucleotide sequences from more than 140, 000 organisms.

Important contributors to the EST data base of the U.S. National Center for Biotechnology (dbEST) and to supporting its public availability include Washington University Genome Sequencing Center (St. Louis, Missouri) through its support by the Howard Hughes Medical Institute; members of the I.M.A.G.E. Consortium; and Merck & Co., Inc. (Whitehouse Station, New Jersey).

The Institute for Genomic Research (TIGR) in Rockville, Maryland, is a not-for-profit research institute with academic partnerships throughout the world. TIGR researchers have completed the genome sequencing of many pathogens. Published EST sequence data are available through the TIGR Web site, `www.tigr.org`.

GeneX Data Base, supported by the U.S. National Center for Genomic Research (NCGR), and ArrayExpress, supported by European Bioniformatics Institute (EBI) in the UK, are compliant with current recommendations of standardization [39, 40].

The Stanford Microarray Data Base (SMD), which is Internet accessible (`http://genome-www5.stanford.edu/`) is implementing annotations developed by the Array SML working group at Stanford University, Laboratory of Patrik O. Brown, Department of Biochemistry and Biophysics.

Institutes of the Human Genome Project: Whitehead Institute for Biomedical Research in Cambridge, Massachusetts; The Wellcome Trust Sanger Institute in Hinxton, Cambs, UK; Baylor College of Medicine in Houston, Texas; Washington University in St. Louis, Missouri; Department of Energy's Joint Genome Institute (JGI) in Walnut Creek, California.

2.5.1 Uses of Microarrays

As a genomic readout, microarrays can serve for many purposes, and novel applications are arising [87]. A common application of microarrays has been the measurement of gene expression, from characterizing cells and processes ([74, 75, 133]) to clinical applications such as tumor classification [20, 107]. Another very common use of microarrays is in genotyping and the measurement of genetic variation [172, 249].

Microarrays have been used to identify the RNA components of various complexes, clarifying on biological mechanisms of RNA translation and transport [87, 138, 230]. Recently, it has been identified complexes of protein and RNA, called P-bodies, are thought to be involved in gene expression by regulating mRNA in the cytoplasm [195]. Microarrays could be used to monitor and characterize the trafficking of cellular RNA through this complex. Changes in DNA copy number at various loci have been implicated in tumorigenesis and cancer. Using comparative genomic hybridization, microarrays have been used to examine aneuploidy and changes in loci copy number in a variety of cell types [197,216]. Microarrays have been used to examine the progress of replication forks as they copy the genome [149], as well as for genome-wide screens of RNA modifying enzymes [121]. The full range of applications is too numerous to mention, improvements and adaptations are continually being made [123].

However, the rapid growth of microarray research over the past few years involves the use of microarrays to identify functional elements in the genome [87]. Expression of a gene in the form of an RNA transcript is one small slice of the biology of a gene. A fundamental aspect of gene expression currently being explored by microarrays is the revelation of control elements in the genome that are responsible for turning genes on and off. Every gene is under the control of a regulatory code. However, this code is largely unknown.

Taken together, the data being collected by tiling arrays for both protein/DNA interactions and identification of novel transcripts in humans are being systematically and jointly analyzed as part of a large consortium termed as the ENCODE Project with the aim of compiling a comprehensive encyclopedia of DNA elements [62, 63]. Microarrays are a fundamental aspect of this effort [87].

Chapter 3

Data Clustering

Data mining is the process of using a variety of data analysis tools to discover patterns and relationships in the data, in such a way that may be used to make valid predictions [113, 114, 163, 178, 190, 208]. The first and simplest analytical step in data mining is to *describe* the data, summarize its statistical attributes, visually review it using charts and graphs, and look for potentially meaningful links among variables (such as values that often occur together). The emphasis in collecting, exploring and selecting the right data is critically important.

However, data description alone can not provide an action plan, accordingly, *a predictive model* most be built based on patterns determined from known results and then, to *test* such a model on results outside the original sample. A good model should never be confused with reality, but it can be a useful guide to understanding our data. The final step of this process is to empirically *verify* the model.

Data mining offers great promise in helping organizations to find out patterns hidden in their data that can be used to predict the behavior of customers, products and processes. Nevertheless, data mining tools need to be guided by users who understand the problem, the data, and the general nature of the analytical methods involved. Realistic expectations can yield rewarding results across a wide range of applications from improving revenues to reducing costs.

Data mining takes advantage of advances in the fields of *artificial intelligence* and statistics [88]. Both disciplines have been working on problems of pattern recognition and classification. These communities have made great contributions to the understanding and application of neural nets and decision trees.

Cluster analysis is an important task within data mining; it deals with dividing a data set into different groups [83]. The goal of clustering is to find groups that are very different from each other, and whose members are very similar to each other. Unlike classification, in cluster analysis it is not known what the clusters will be when one starts, or which attributes in data will be clustered. After one has found clusters that reasonably segment a database, these clusters may then be used to classify new data. Clustering is a way to segment data into groups that are not previously defined, whereas classification is a way to segment data by assigning it to groups that are already defined.

Graphing and visualization tools are of vital aid in data preparation and their importance to effective data analysis can not be overemphasized. Data visualization most often provides knowledge leading to new insights and success. Some of the common and very useful graphical displays of data are histograms or box plots that show distributions of values. It may also want to look at scatter plots in two (or in three) dimensions of different pairs of variables. The ability to add a third, overlay variable greatly increases the usefulness of some types of graphs.

This chapter introduces the data cluster problem and outlines issues as representation, types, normalization and visualization of data. We also explain the different types of methods and algorithms of data clustering, giving a tree of clustering method classification. This section concludes through of a pseudocode showing the steps for the single-link and complete-link clustering algorithms. The following section is dedicated to introduce the cluster validation subject and enumerate the cluster validation measures most commonly used in cluster analysis. At the end of this chapter, we give a review of clustering and visualization techniques currently involved in the analysis of DNA microarray data.

3.1 Cluster Analysis

Classifying data according to their similarity degree is one of the main processes of data mining. The organization of objects in affinity groups is one way of knowledge discovery, being a key factor in *machine learning* [8, 10].

Cluster analysis is the formal study of methods and algorithms for grouping or classifying objects. An object is described either by a set of measurements or by relationships between the object and other objects. Cluster analysis does not use category labels that tag objects with prior identifiers. The absence of category labels distinguishes cluster analysis from discriminant analysis, pattern recognition and decision analysis.

The goal of cluster analysis is simply to find a convenient and valid organization of the data, not to establish rules for separating future data into categories. Clustering algorithms are focused on discovering structures in the data. This way, a cluster can be understood as a set of similar objects, collected or grouped together.

The most widely accepted definitions of cluster according to [91] can be described as follows:

1. A cluster is a set of entities which are alike, and entities from different clusters are not alike.

2. A cluster is an aggregation of points in the test space such that, the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it.

3. Clusters may be described as connected regions of a multi-dimensional space containing a relatively high density of points, separated from other such regions by a region containing a relatively low density of points.

The last two definitions assume that the objects to be grouped represent points in the measurement space. Hence, it is not hard to give a functional definition of a cluster; however, it is complex to give an operational definition of a cluster. Because objects can be grouped into clusters according to different purposes; moreover, data can disclose clusters of different shapes and sizes. Even more complex, cluster membership can change over time, and the number of clusters often depends on the resolution with which the data are seen. Therefore, the key problem in identifying clusters in the data is to specify the proximity between the data and how to measure it. As expected, the notion of proximity is domain dependent.

Cluster analysis is one component of exploratory data analysis, which means sifting through data to make sense out of measurements by whatever means are available.

The information gained about a set of data from a cluster analysis should suggest new experiments and provide fresh insight into the subject matter.

3.2 Data Representation

The first condition that most have a rational application of cluster analysis is an appreciation for the basic factors required to represent the data. Clustering algorithms are closely linked to data type, which implies that if the factors such as scale, normalization, and types of proximity measures are not understood, one can make mistakes interpreting the results of them.

Cluster analysis is an exploration tool of data, due to that, it is necessary to use techniques for visualizing data. The most direct visualization is a two-dimensional plot showing the objects to be clustered as points. Multivariate data can not always be suitably represented in the plane but when valid, such a visualization allows us to verify the results of the clustering algorithms.

The intrinsic (or topological) dimensionality of the data dictates the smallest number of factors needed to represent them. Clustering algorithms group objects (or individuals) based on *proximity indices* between pairs of objects. A set of objects (a data set) comprises the raw data for a cluster analysis and can be described by two standard formats: a data matrix and a proximity matrix.

The data matrix is formed by the n objects that will be grouped as rows, and if these objects consist of d measurements (attributes or scores), then the order of this matrix is $n \times d$. That is, each object is a vector of d *features*. The d features are usually pictured as a set of orthogonal axes and so, the n objects are embedded into d-dimensional space.

A cluster can be visualized as a collection of data which are close to one another or which satisfy some spatial relationships. The task of a clustering algorithm is to identify such natural groupings in spaces of many dimensions. Although visual perception is limited to three dimensions, one must be careful not to think automatically of clustering problems as two- or three-dimensional. The real benefit of cluster analysis is to organize multidimensional data where visual perception fails.

Turning to the proximity matrix, clustering methods require an index of proximity, or alikeness, or affinity, or association be established between pairs of objects. This index can be computed from a data matrix or can be formed from raw data. A proximity matrix $\mathfrak{D} = [d(i, j)]$ accumulates the pairwise indices of proximity in a matrix in which each row and column represents an object. We assume that a proximity matrix is symmetric since all pairs of objects have the same proximity value, independently of the order of their objects. Moreover, the main diagonal of it is 0, because we deal with proximities between one object and itself.

A proximity index is either a similarity or dissimilarity (for example, correlation coefficient and Euclidean distance, respectively). The more the ith and jth objects resemble one another, the larger a similarity index and the smaller a dissimilarity index. A thorough review of measures of association and their interrelationships is provided in [22]. According to that, a proximity index between the ith and jth objects, denoted by $d(i, j)$ must satisfy the following three properties:

1. For a dissimilarity: $d(i, i) = 0$, for all object i.
   For a similarity: $d(i, i) \geq \max_j d(i, j)$, for all object i;
2. $d(i, j) = d(j, i)$, for all pair of objects (i, j);
3. $d(i, j) \geq 0$, for all pair of objects (i, j).

All clustering algorithms generally use either Euclidean distance or correlation coefficient of Pearson as the proximity measure.

3.3 Data Types and Scales

Understanding the data type and the scale used on the data is of vital importance for selecting a clustering algorithm to solve a problem. Data type refers to the degree of quantization in the data. A single feature can be typed as binary, discrete, or continuous. Generally, all features are discrete; however, it is often convenient to think of a feature value as a point on the real line that can take on any real value in a fixed range of values. Such a feature is called *continuous*.

The second trait of a feature and of a proximity index is the data scale, which indicates the relative significance of numbers ([22] outlines a categorization of data types and data scales appropriate for cluster analysis). Data scale can be divided into *qualitative* (nominal or ordinal) and *quantitative* (interval or ratio) scales:

1. The *nominal* scale is not really a scale at all because numbers are simply used as names;
2. In the *ordinal* scale, the numbers have meaning only in relation to one another;
3. The separation between numbers has meaning on an *interval* scale. Thus, a unit of measurement exists, and the interpretation of the numbers depends on this unit;
4. The strongest scale is the ratio scale, on which numbers have an absolute meaning. This implies that an absolute zero exists along with a unit of measurement, so the ratio between two numbers has meaning.

Recognizing type and scale is important in both forming proximity indices and interpreting the results of a cluster analysis. Several formats, types and scales for data are summarized in Figure 3.1.

3.4 Data Normalization and Visualization

In practice, it is seldom to analyze a data set (raw data with a defined distance) without performing any normalization process. Some normalization is usually employed based on the requirements of the analysis. Preparing the data for a cluster analysis requires some sort of normalization that takes into account the measure of proximity. Such is the case of the Euclidean distance, which is a popular index of dissimilarity, but it implicitly assigns more weighting to features with large ranges than to those with small ranges. Scaling one feature in kilometers and a second feature in centimeters makes the second feature numerically overpower the first. Hence, a normalization process should be able to solve these problems. Several normalization schemes that remedies some of these problems are presented in [8, 35, 45, 100, 112, 171].

However, special care must be taken when normalization is applied since normalization or scaling is not always desirable. For example, if the spread among the objects is due to the presence of clusters, the normalization can change the inter-point distances and can alter the separation between natural clusters.

On the other hand, for data visualization, the concept of data projection is used to reduce data dimensionality, and visually represent the data. In general terms,

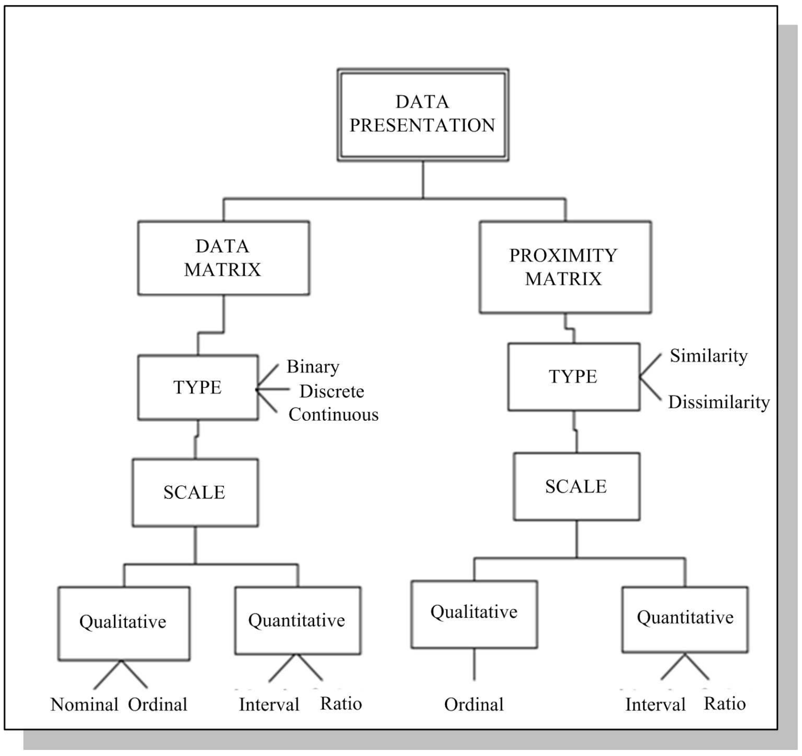


Figure 3.1: Formats, types and scales for representing data.

the algorithms of data projection map a set of n d-dimensional objects onto an m-dimensional space with $m < d$. The main motivation for studying projection algorithms in the context of cluster analysis is to allow visual examination of multivariate data. When a reasonably accurate two-dimensional representation of a set of objects can be obtained, one can cluster by eye and qualitatively validate conclusions drawn from clustering algorithms.

There exist two forms of making data projections, that is, linear and nonlinear projection. The first, expresses the m new features as linear combinations of the original d features. Linear projection algorithms are relatively simple to use, tend to preserve the character of the data, and have well-understood mathematical properties. The type of linear projection used in practice is influenced by the availability of category information about the objects in the form of labels on the objects. So that the two types of projections most commonly used are: *principal component analysis* ([142]) when no category information is available and *discriminant analysis* ([132]) when category labels are available.

The inability of linear projections to preserve *complex data structures* (data lie on a curved surface) has made nonlinear projections more popular in recent years. Most nonlinear projection algorithms are based on maximizing or minimizing a function of a large number of variables. This optimization problem is data dependent and does not involve an explicit mapping function. Thus, a change in the number of objects requires recomputing the entire projection.

Nonlinear projection algorithms can be derived from two points of view, depending on the prior information available about the patterns. If category information is known, the aim is to find a nonlinear projection that reduces dimensionality yet maximizes separability between categories. In the absence of category information, the goal is to project the data onto a low-dimensional space so as to retain as much structure as possible. However, nonlinear projection algorithms are expensive to use, so several heuristics must be employed to reduce the search time for the optimum solution. So that the best principal component projection could be used as the starting configuration for a nonlinear mapping algorithm.

For exploratory data analysis, we seek two-dimensional projections to visually perceive the structure present in the data. The selection of a nonlinear projection algorithm for exploratory data analysis is affected by the form of the available data. Additionally, while a projection algorithm seeks two coordinates that preserve structural information contained in the original d features, a graphical representation tries to preserve this information exactly by using all the features as attributes of graphical representation for the objects.

3.5 Methods and Algorithms for Clustering

Cluster analysis is the process of classifying objects into subsets in such a way that, these subsets have meaning in the context of a particular problem. The objects are thereby organized into an efficient representation that characterizes the population being sampled [8, 9].

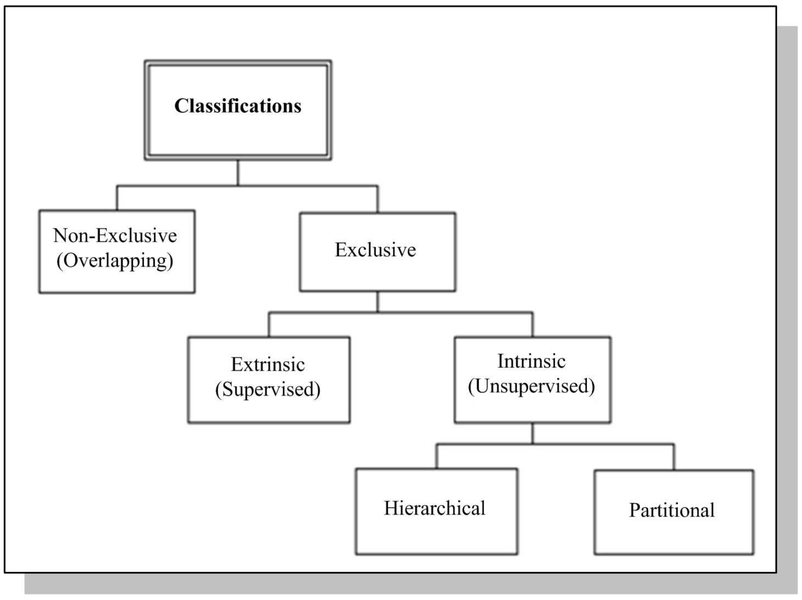


Figure 3.2: Tree of classification types.

A clustering is a type of classification imposed on a finite set of objects, where the relationship between objects is represented in a proximity matrix in which rows and columns correspond to objects. In this case, the proximity matrix is the one and only input to a clustering algorithm.

Since a clustering is a special kind of classification [148], a tree of data classifier kinds, suggested in [161] is shown in Figure 3.2. Each leaf in the tree defines a different genus of classification problem. The nodes in the tree are defined below.

1. *Exclusive vs. nonexclusive.* An exclusive classification is a partition (in mathematical terms) of the data set. Each object belongs to exactly one subset, or a cluster. Nonexclusive, or overlapping, classification can assign an object to several classes. This research deals only with exclusive classification.

2. *Intrinsic vs. extrinsic.* An intrinsic classification uses only the proximity matrix to perform the classification. Intrinsic classification is called *unsupervised learning* in pattern recognition because no category labels denoting a priori partition of

the objects are used. Extrinsic classification uses category labels on the objects as well as the proximity matrix (*supervised learning*). The problem is then to establish a discriminant surface that separates the objects according to category. We are concerned only with intrinsic classification due to it is the essence of cluster analysis.

3. *Hierarchical vs. partitional.* Exclusive, intrinsic classifications are subdivided into hierarchical and partitional classifications by the type of structure imposed on the data. A hierarchical classification is a nested sequence of partitions, whereas a partitional classification is a single partition of a data set. Thus a hierarchical classification is a special sequence of partitional classifications. The term *clustering* will be used for an exclusive, intrinsic, partitional classification and the term *hierarchical clustering* for an exclusive, intrinsic, hierarchical classification. In this research, we deal only with hierarchical clustering, which is our topic of research.

3.5.1 Hierarchical Clustering

A hierarchical clustering method is a procedure for transforming a proximity matrix into a *sequence of nested partitions*. A hierarchical clustering algorithm is the specification of steps for performing a hierarchical clustering. It is often convenient to characterize a hierarchical clustering method by writing down an algorithm, but the algorithm should be separated from the method itself.

Before turning to see the different hierarchical clustering methods and algorithms, we define the mathematical structure that characterizes a hierarchical clustering. We first denote the data set and its partition. The n objects to be grouped into different subsets are represented by a set $\mathfrak{P} = \{x_1, x_2, \ldots, x_n\}$. Each x_i is the ith object of $\mathfrak{P}$. Then, a partition $\mathfrak{C}$ (called clustering) of $\mathfrak{P}$, breaks $\mathfrak{P}$ into subsets $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m\}$ (called clusters), holding the following conditions:

$$\mathcal{C}_i \cap \mathcal{C}_j \neq \phi, \quad (\forall i, j \in [1, m] \wedge i \neq j)$$

and

$$\bigcup_{i=1}^{m} \mathcal{C}_i = \mathfrak{P}.$$

We now can define nested partition sequence as follows: partition $\mathfrak{B}$ is nested into partition $\mathfrak{C}$ if every component in $\mathfrak{B}$ is a subset of a component in $\mathfrak{C}$. That is, $\mathfrak{C}$ is formed by merging components of $\mathfrak{B}$. A hierarchical clustering is a sequence of partitions in which each partition is nested into the next partition in the sequence.

Several algorithms can be proposed to express the same exclusive, intrinsic classification. One frequently uses an algorithm to express a clustering method and then, examines various computer implementations of the method. An *agglomerative* hierarchical classification places each object in its own cluster and gradually merges these atomic clusters into larger and larger clusters until all objects are in a single cluster. *Divisive* hierarchical classification reverses the process by starting with all objects in one cluster and subdividing into smaller pieces. Thus this option corresponds to choose of procedure rather than the type of classification.

An agglomerative algorithm starts with the disjoint clustering, which places each of the n objects in an individual cluster. The clustering algorithm being used dictates how the proximity matrix should be interpreted to merge two or more of these trivial clusters. Thus, it nests the trivial clustering into a second partition. The process is repeated to form a sequence of nested clusterings, where the number of clusters decreases as the sequence progresses until a single cluster containing all n objects. A divisive algorithm performs the task in the reverse order.

A hierarchical structure of clustering facilitates the visual exploration of data. A hierarchical clustering uses a special structure of tree that provides a suitable visual representation of clustering hierarchy, called *dendrogram*. A dendrogram consists of layers of nodes, each representing a cluster. Lines connect nodes representing clusters which are nested into one another. Cutting off a dendrogram horizontally creates a clustering. Figure 3.3 provides a simple example.

The level, or proximity value, at which a clustering is formed, can also be recorded. If objects are represented as patterns, or points in a space, the centroids of the clusters can be important, as well as the spreads of the clusters.

3.5.2 Algorithms for Single-link and Complete-link Methods

There are many agglomerative algorithms, which only differ in their definition of between-clusters dissimilarity. The most used criteriums are: minimum distance (single-link), average distance (mean-link) and maximum distance (complete-link), [10]. These

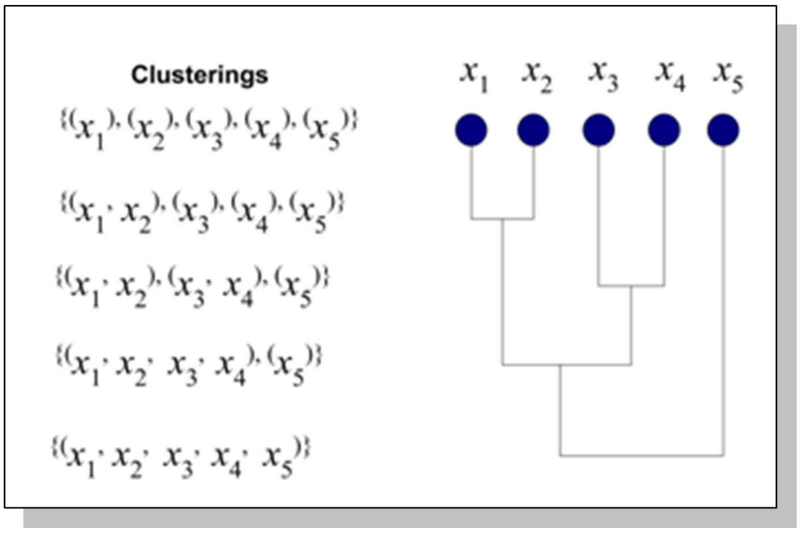

Clusterings
{(x1), (x2), (x3), (x4), (x5)}
{(x1, x2), (x3), (x4), (x5)}
{(x1, x2), (x3, x4), (x5)}
{(x1, x2, x3, x4), (x5)}
{(x1, x2, x3, x4, x5)}
x1
x2
x3
x4
x5

Figure 3.3: Example of a dendrogram.

three criteriums are useful in different types of applications. That is, single-link can be appropriated for applications where one really wants to find elongated clusters. Mean-link is aimed at finding roughly ball-shaped clusters. Complete-link tends to produce very compact clusters of a small diameter. The resulting clusters are not necessarily well separated, which means that all clusters are very close.

The single-link algorithm refers to the shortest distance between objects of two clusters. So that mean-link and complete-link algorithms refer to the average distance and the greater distance between objects of two clusters, respectively. These algorithms assume that the proximity matrix has no repeated distances. Thus, they produce a single nested clustering sequence that can be shown through its associated dendrogram.

Below there is a pseudocode (Algorithm 3.5.3) for single-link and complete-link algorithms (mean-link can be deduced from it) [131], where clusterings are enumerated as $0, 1, \ldots, (n-1)$ and the mth clustering ($\mathfrak{C}_m$) has $n - m$ clusters:

$$\mathfrak{C}_m = \{\mathcal{C}_{m1}, \mathcal{C}_{m2}, \ldots, \mathcal{C}_{m(n-m)}\}.$$

3.5.3 Agglomerative algorithm for single-link and complete-link methods

1. Set $m \leftarrow 0$. Form the disjoint clustering,

$$\mathfrak{C}_0 = \{(x_1), (x_2), \ldots, (x_n)\}.$$

2. **a)** To build the next clustering $m+1$ by the single-link method. Function Q_s defines for all pairs (r,t) of clusters in the current clustering as follows:

   $Q_s(r,t) = \min\{d(i,j)/i \in \mathcal{C}_{mr} \wedge j \in \mathcal{C}_{mt}\}$.

   Clusters $\mathcal{C}_{mp}$ and $\mathcal{C}_{mq}$ are merged to form the next clustering in the single-link hierarchy if, $\mathcal{Q}_s(p,q) = \min\{Q_s(r,t)\}$.

3. **b)** Function $\mathcal{Q}_c$ is used to build clustering $m+1$ by the complete-link method and is defined for all pairs (r,t) of clusters in the current clustering:

   $Q_c(r,t) = \max\{d(i,j)/i \in \mathcal{C}_{mr} \wedge j \in \mathcal{C}_{mt}\}$.

   Clusters $\mathcal{C}_{mp}$ and $\mathcal{C}_{mq}$ are merged to form the next clustering in the complete-link hierarchy if, $\mathcal{Q}_c(p,q) = \min\{Q_c(r,t)\}$.

4. Set $m \leftarrow m+1$ and repeat step 2. Continue until all objects are a single cluster.

Single-link clusters are characterized as maximally connected subgraphs whereas complete-link clusters are cliques, or maximally complete subgraphs. In contrast to the agglomerative method, the divisive approach offers a distinct advantage in that most users are concerned in the main structure of their data, consisting of a few large clusters, rather than in a detailed description of the individual points. Divisive analysis starts with the main chunks. In the first step it splits the data into two parts and then goes on by dividing them further into smaller parts.

Divisive analysis poses some computational problems, at last in principle. All possible divisions of the data into two subsets taken from the first step of the algorithm could make infeasible this issue. Hence, many authors have restricted their attention to the agglomerative approach. Therefore, divisive clustering algorithms are not generally available, [10]. Several algorithms for agglomerative and divisive hierarchical clustering have been used. Namely, agglomerative algorithms: *Agnes* and *Eisen* in [10, 86] respectively. As divisive algorithms: *Diana, TSVQ* and *HybridHclust* in [10, 55, 174] respectively.

One final aspect to consider is that, although hierarchical clustering methods generate dendrograms, which are an appropriate way to display information. It is difficult comparing two dendrograms. This motivates the development of methods (and visualizations) for automatically isolating significant clusters. On the other hand, clustering methods have the nasty habit of creating clusters in data even when no natural clusters exist. So, hierarchies and clusterings must be viewed with extreme suspicion [8].

3.6 Cluster Validation

As stated in precedent sections, there are different hierarchical clustering algorithms with different criteriums for building their clusters. According to such criteriums, we obtain different results from the partition of a data set. Thus, it is important to compare results of several clustering algorithms and choose the one that best fits the true data distribution. Nevertheless, this is not an obvious task.

Cluster validation refers to procedures that evaluate the results of cluster analysis in a quantitative and objective fashion. Hierarchies, clusterings and clusters are some times justified by ad hoc methods based on the application area.

Thus, cluster validation is considered as the process of assessing the quality and reliability of the cluster sets derived from various clustering processes [137]. Generally, cluster validity has three aspects. First, the quality of clusters can be measured in terms of *homogeneity* (or *compactness*) and *separation* on the basis of the definition of a cluster. The second aspect relies on a given *ground truth* (or *reference partition*) of the clusters. The ground truth could come from domain knowledge. Cluster validation is based on the agreement between clustering results and the ground truth. And the third aspect of cluster validity focuses on the reliability of the clusters or the likelihood that the cluster structure is not formed by chance.

It is easy to propose indices of cluster validity. It is very difficult to fix thresholds on such indices that define when the index is large or small enough to be unusual. Statistical methods provide a framework for rationally deciding how large is “large” and how small is “small” [8]. Several textbooks in statistics cover this material, such as in [61, 248].

3.6.1 Homogeneity and Separation

There exist several definitions for the homogeneity of clusters which measure the similarity of objects in cluster $\mathcal{C}$. Namely:

$$\mathcal{H}_1(\mathcal{C}) = \frac{\sum_{i \neq j} Similarity(i,j)}{|\mathcal{C}| \cdot (|\mathcal{C}| - 1)}, \quad (\forall i, j \in \mathcal{C}) \tag{3.6.1}$$

An alternative definition evaluates the homogeneity with respect to the *centroid* of cluster $\mathcal{C}$ as follows:

$$\mathcal{H}_2(\mathcal{C}) = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} Similarity(i, \bar{O}), \tag{3.6.2}$$

where $\bar{O}$ is the centroide of $\mathcal{C}$. Cluster separation is analogously defined from various perspectives to measure the dissimilarity between two clusters $\mathcal{C}_1, \mathcal{C}_2$:

$$\mathcal{S}_1(\mathcal{C}_1, \mathcal{C}_2) = \frac{\sum_{i \in \mathcal{C}_1 \wedge j \in \mathcal{C}_1} Similarity(i,j)}{|\mathcal{C}_1| \cdot |\mathcal{C}_2|} \tag{3.6.3}$$

and,

$$\mathcal{S}_2(\mathcal{C}_1, \mathcal{C}_2) = Similarity(\bar{O}_1, \bar{O}_2) \tag{3.6.4}$$

homogeneity and separation of a clustering $\mathfrak{C} = \{\mathcal{C}_1, \mathcal{C}_1, \ldots, \mathcal{C}_m\}$ are defined ([217]) as:

$$\mathcal{H}_{ave} = \frac{1}{m} \sum_{\mathcal{C}_i \in \mathfrak{C}} |\mathcal{C}_i| \cdot \mathcal{H}_2(\mathcal{C}_i), and \tag{3.6.5}$$

$$\mathcal{S}_{ave} = \frac{1}{\sum_{\mathcal{C}_i \neq \mathcal{C}_j} |\mathcal{C}_i| \cdot |\mathcal{C}_j|} \sum_{\mathcal{C}_i \neq \mathcal{C}_j} |\mathcal{C}_i| \cdot |\mathcal{C}_j| \cdot \mathcal{S}_2(\mathcal{C}_i, \mathcal{C}_j) \tag{3.6.6}$$

3.6.2 Silhouette Width

Silhouette width ([207]) is a composite index, reflecting the compactness and separation of clusters. As its predecessors, it can also be applied on different metrics. Then for an object i, its silhouette width $s(i)$ is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}. \tag{3.6.7}$$

$a(i)$ is the average dissimilarity from object i to the remaining objects of the same cluster. $b(i)$, average dissimilarity from i to all objects of the closest cluster. The clustering silhouette width is defined as the average of all $s(i)$. This index reflects the overall quality of a clustering. Moreover, it is followed from formula (3.6.7) that $-1 \leq s(i) \leq 1$. If silhouette value is close to 1, then sample is well-clustered and it was assigned to a very appropriate cluster. Hence, the largest overall average silhouette indicates the best clustering.

3.6.3 Agreement with Reference Partition

If the ground truth of the cluster structure of the data set is available, then it is possible to test the performance of a clustering process by comparing the clustering results with the ground truth.

Given the clustering results $\mathfrak{C} = \{\mathcal{C}_1, \mathcal{C}_1, \ldots, \mathcal{C}_m\}$ and the ground truth $\mathfrak{T} = \{\mathcal{T}_1, \mathcal{T}_1, \ldots, \mathcal{T}_m\}$, some common indices ([111, 225]) have been defined to measure the degree of similarity between $\mathfrak{C}$ and $\mathfrak{T}$:

$$\text{Rand index: } RI = \frac{n_{11} + n_{00}}{n_{11} + n_{10} + n_{01} + n_{00}}, \tag{3.6.8}$$

$$\text{Jaccard coefficient: } JC = \frac{n_{11}}{n_{11} + n_{10} + n_{01}}, \tag{3.6.9}$$

$$\text{Minkowski measure: } MM = \sqrt{\frac{n_{10} + n_{01}}{n_{11} + n_{01}}}. \tag{3.6.10}$$

Each n_{ij} is given from two matrices: an $n \times n$ binary matrix M, where n is the number of objects, $M_{ij} = 1$ if objects i, j, belong to the same cluster, and $M_{ij} = 0$ otherwise. Similarly, the binary matrix N is built for the ground truth $\mathfrak{T}$. Then, values n_{ij} are set as follows:

- n_{11} is the number of object pairs (i, j), where $M_{ij} = 1$ and $N_{ij} = 1$;
- n_{10} is the number of object pairs (i, j), where $M_{ij} = 1$ and $N_{ij} = 0$;
- n_{01} is the number of object pairs (i, j), where $M_{ij} = 0$ and $N_{ij} = 1$;
- n_{00} is the number of object pairs (i, j), where $M_{ij} = 0$ and $N_{ij} = 0$.

The Rand index and the Jaccard coefficient (JC $\in (0, 1]$, RI $\in [0, 1]$) measure the extent of agreement between $\mathfrak{C}$ and $\mathfrak{T}$, while the Minkowski measure (MM ≥ 0) illustrates the proportion of disagreements to the total number of object pairs (i, j), where i, j belong to the same set in $\mathfrak{T}$. Note that the optimal index selection in a problem is application dependent.

3.7 Clustering and Visualization Methods for DNA Microarrays

This section outlines different approaches of clustering methods used in cluster analysis of DNA microarray data, and the existing visualization techniques to represent these data. A set of visualization tools of clustering results from DNA microarray data is given as final part of this section. These tools include a part of the existing visualization components to represent data, and so, a comparison between them is made, where tool 3D-VisualCluster product of our framework is also compared.

3.7.1 Gene-Based Clustering Approaches

For each clustering approach, we introduce the basic idea of the clustering process, and then explain some features of the given algorithm. For more details see [137] and the remainder of the proposed bibliography.

K-Means

The K-means algorithm is a partitional clustering method [184], which from a determined number K, partitions a given data set into K disjoint clusters according to optimize $\sum_{i=1}^{K} \sum_{O \in C_i} \mid O - \mu_i \mid^2$, where C_i represents each cluster of the clustering, and μ_i is the centroid of C_i computing from the mean of objects in it. The aim is to minimize the sum of the squared distances of objects from their cluster centers.

The K-means algorithm usually converges in a small number of iterations, however, it also presents several drawbacks on the clustering task for DNA microarray data. The number of clusters for DNA microarray data is unknown in advance, therefore, some users run the algorithm many times for different values of K, taking out the K whose partition is minimum at all. The problem is that for large data sets of thousands of genes, this strategy may not be practical. On the other hand, K-means forces each gene into specific cluster, which may cause it to be sensitive to noise [218, 223], being a problem, considering a huge amount of noise presents in DNA microarray data. More recent approaches of clustering algorithms has been proposed in [120,180,200] to overcome the drawbacks of K-means. These algorithms usually depend of some global parameters to check the quality of resulting clusters. However, the qualities of clusters for gene expression data may vary widely. Hence, the choice of the appropriate globally constraining parameters be generally a hard problem.

Hierarchical Clustering Approaches

Unlike the partitional approach, hierarchical clustering generates a hierarchy of nested clusters that can be represented graphically trough a dendrogram. The branches of the dendrogram apart of recording the formation of clusters, indicate also the similarity between them. In [86] an agglomerative algorithm called UPGMA (Unweighted Pair Group Method with Arithmetic mean, it is also known as Eisen method) and a graphical method to represent the clustered data were applied to DNA microarray analysis. In this method, the gene expression matrix is displayed as a heat map, where each expression level in the matrix has assigned a color (an intensity level of a given color) according to the measured fluorescence ratio, and the rows of the matrix are reordered based on the dendrogram structure of UPGMA. The final result is a colored table (a cluster image) representing gene groups that share similar expression patterns over multiple conditions.

[21] used a divisive approach called DAA (the Deterministic-Annealing Algorithm) to split the genes [205, 206]. Initially, this algorithm randomly chooses two cluster centroids μ_1, μ_2 and represents the expression pattern of a gene i as a vector $\vec{g}_i$. According to this, the probability that a gene $\vec{g}_i$ belongs to a cluster j was assigned

stating a two-component Gaussian model:

$$P_j(\vec{g}_i) = \exp(-\beta \mid \vec{g}_i - \mu_j \mid^2) \diagup \sum_j \exp(-\beta \mid \vec{g}_i - \mu_j \mid^2).$$

After this, the cluster centroids are recalculated as:

$$\mu_j = \sum_i \vec{g}_i P_j(\vec{g}_i) \diagup \sum_i P_j(\vec{g}_i).$$

An iterative process called the EM algorithm was then applied to solve P_j and μ_j. The whole data set was recursively split until each cluster contained only one gene.

The Agnes algorithm in [10] (*Agglomerative nesting*) builds a hierarchy of clusterings. At first, each data is a small cluster by itself. Clusters are merged until only one large cluster remains containing all data. At each stage the two nearest clusters are combined to form one larger cluster. The Diana algorithm in [10] (*Divisive analysis*) performs the task in the reverse order, starting from one large cluster containing all data. Clusters are divided until each cluster contains only a single data. At each stage, the cluster with the largest diameter is selected to be split.

The TSVQ algorithm in [174] (*Tree-structured Vector Quantization*) builds a divisive hierarchical clustering, so the data must be recursively subdivided into two clusters. Hence, 2-means is used to find a subdivision. The HybridHclust algorithm in [55] (*Hybrid Hierarchical clustering*) is a divisive hierarchical clustering where TSVQ is applied to data with constraint that mutual clusters cannot be divided. Within each mutual cluster, TSVQ is re-applied to yield a top-down hybrid in which mutual cluster structure is retained. Since HybridHclust is based on TSVQ, it implicitly uses squared Euclidean distance between data.

Hierarchical clustering provides a graphic representation allowing users a thorough inspection of the whole data set and obtain an initial impression of the distribution of data. However, the conventional agglomerative approach suffers from a lack of robustness [231]. For small perturbations of the data set may greatly change the structure of the hierarchical dendrogram. Furthermore, the greedy nature of both approaches, agglomerative and divisive, prevents the refinement of the clusterings in the dendrogram.

Graph-Theoretical Approaches

From a data set it is possible to build a graph $\mathcal{G}(V, E)$ called a proximity graph. Every data point corresponds to a vertex (in V) and each pair of objects is connected by an edge (in E) with weight assigned according to the proximity matrix. In some cases the proximity matrix is mapped only to either 0 or 1 on the basis of some threshold, and edges only exist when the proximity between the objects is 1. Graph-theoretical clustering converts the problem of clustering into graph theoretical problems as finding minimum cut or maximal cliques in the proximity graph $\mathcal{G}$.

- *CLICK (CLuster Identification via Connectivity Kernels):* CLICK in [217] seeks to identify highly connected components in the proximity graph as clusters. It starts from the probabilistic assumption that after standardization, similarity values between elements are normally distributed. From this assumption, it defines the weight of an edge (i, j) as the probability that vertices i and j are in the same cluster. CLICK then iteratively finds the minimum cut in the proximity graph. From here, it recursively splits the data set into a set of connected components from the minimum cut.

  In experiments carried out in [217], CLICK shown better quality in terms of homogeneity and separation. However, it faces the problem of building highly unbalanced partitions. Moreover, for gene expression data where two gene clusters can be highly intersected each other, CLICK may not be capable to separately identify them since both clusters would be identified as one highly connected component [137].

- *CAST:* In [33] a corrupted clique graph data model was introduced. This model assumes that the data set comes from a cluster structure affected by *contamination* through random errors caused from the complex process of DNA microarray data preparation. In fact, it assumes that the true clusters are those represented by a clique graph $\mathcal{H}$, which is a disjoint union of complete subgraphs with each clique corresponding to a cluster. The similarity graph $\mathcal{G}$ is obtained from $\mathcal{H}$ by flipping each edge (nonedge) with a given probability. Hence, the clustering problem is equivalent to achieve the original clique graph $\mathcal{H}$ from the corrupted graph $\mathcal{G}$ with as few flips (errors) as possible.

CAST (*Cluster Affinity Search Technique*) specifies the desired cluster quality through a parameter t and applies a searching process to identify qualified clusters one at a time. Hence, it does not require that the user specifies the number of clusters. Furthermore, it has an effective outlier treatment. However, it is known that CAST faces the difficulty of determining a “good” value for the global parameter t.

Model-Based Clustering

The approach of model-based clustering proposes a statistic model to describe the cluster structure of gene expression data [95,103,183,255]. The data set is assumed to come from a finite mixture of underlying probability distributions, with each component corresponding to a different cluster. The goal is to estimate the parameters given in the sets Θ and Γ that maximize a likelihood $L_{mix}(\Theta,\Gamma)$. The parameters Θ and Γ are estimated with the EM algorithm, which in the first stage, estimates conditionally the hidden parameters Γ from the data with the current Θ. In the second stage, parameters Θ are estimated in order to maximize the likelihood of complete data given the estimated hidden parameters. At the end of the algorithm, each object is assigned to the cluster (component) having the maximum conditional probability.

An advantage of this approach is the estimated probability that an object belongs to given cluster. As gene expression data are typically highly connected, it model can detect when a single gene is highly correlated with more than one cluster. Hence, the probabilistic feature of model-based clustering is suitable for DNA microarray data. However, this approach as the previous ones, starts from the assumption that the data set holds a specific distribution, which does not have to be true. Therefore, there is currently no well established model to represent gene expression data.

Density-Based Hierarchical Approach (DHC)

The authors in [135] proposed a clustering algorithm called DHC (a density-based, hierarchical clustering method) to capture coexpressed gene groups from gene expression data. DHC is focused on the notions of *density* and *attraction* of objects in a multidimensional space. As basic idea, a cluster is assumed as a high-dimensional dense region, where data are attracted to each other. At the central part (core), data are crowded closely with each other implying high density. At the boundary region of the cluster, data are relatively sparsely distributed, being attracted by the core part of the dense region.

DHC performs building a cluster structure of the data set, where firstly, an attraction tree is built to represent the relationship between the data in the dense region. After that, it summarizes the cluster structure in the attraction tree into a density tree, where each node represents a dense region. DHC starts considering the whole data set as a large dense region represented by the root node of the density tree. This dense region is then split into several sub-dense regions represented as child nodes of the root node. The process is repeated on these new regions until each sub-dense region be a single cluster.

The DHC approach detects in effective form, groups of coexpressed genes (high density region) from noise (low density region), implying robustness in the noise treatment for DNA microarray data. Moreover, it is also convenient to detect highly connected gene regions. However, the process of computing dense regions could be costly for large data sets.

3.7.2 Visualization Techniques for DNA Microarrays

This subsection outlines several visualization techniques useful in cluster analysis and compares several tools that combine visualizations of DNA microarray data. As was previously studied, cluster analysis on a data domain can imply three stages [115], for which visualization techniques can disclose knowledge, that is:

- *Pre-processing:* Feature selection, normalization, selection of distance function.
- *Cluster analysis:* Selection of algorithm, selection of algorithm parameters, application of algorithm.
- *Cluster validation:* Selection and application of validation techniques.

The aim of visualization techniques in the domain of gene expression data is to explain or validate clustering results as well as to check the use of cluster validation techniques in the analysis of post-genomic. Therefore, in general terms, the application of visual analytics techniques (*the science of analytical reasoning facilitated by* *interactive visual interfaces* [240]) describes an iterative process that involves collecting information, data preprocessing, knowledge representation, interaction and decision making. The ultimate goal is to gain insight into the problem at hand, described by vast amounts of scientific, forensic or business data from heterogeneous sources. To achieve this goal, visual analytics combines the advantages of machines with strengths of humans [146, 147, 167].

Verification of criteriums or hypotheses can be done via visual data exploration. However, it may also be accomplished by automatic techniques from statistics or machine learning. Visual data exploration usually allows faster data exploration and often provides better results, especially in cases where automatic algorithms fail [14]. Usually, this kind of data exploration allows a three-step process: overview, zooming and filtering and then accessing details on demand [177, 220].

Visual analytics is being very useful in Bioinformatics ([26,191]) and particularly, in our case, DNA microarray data analysis [35, 49, 100, 227]. The application of this technique to *data mining* ([113, 190]) of gene expression data can provide knowledge of processes taking place at the cellular level [137]. Visual analytics can also discover knowledge about the variety of available clustering algorithms, for which biologists face problems selecting the most appropriate algorithm for a given gene expression data set, since no single algorithm is the best in every aspect [137].

Visualization Components

Dimensionality reduction algorithms ([93,98,245]) can provide a first inspection of data distribution through a representation in a 2D or 3D scatter plot. Thus, these visualization components as scatter plots are an additional alternative to discover knowledge from the data. However, the performance of this visualization alternative has not been fully exploited. In contrast to the above, the most used visualization component to display DNA microarray data is the combination of dendrogram and head map, introduced in [86]. This component shows the gene expression matrix as a table of intensity of colors (usually, red=up-regulation, green=down-regulation, black=constitutive expression) proportional to differential gene expression (head map). The resulting heap map is reordered according to the dendrogram generated by a given clustering method; the aim is to disclose grouping structures present in the data and dictated by the dendrogram.

Parallel coordinates are other geometric device where genes can be displayed in high dimensional spaces, particulary for more than three dimensions. A representation on parallel coordinates is able to diagnose one-dimensional features such as marginal densities, two-dimensional features such as correlations and nonlinear structures and multidimensional features such as clustering and hyperplanes, [15,36,96,139,246]. Note that the visual validation of a cluster through this component is stricter than the previous ones, providing a mean of validation of visual components of DNA microarray data. Keep in mind that connecting multiple visualizations through interactive linking, provides more information than considering the component visualizations independently.

Visualization Tools

Visual validation approaches applied to cluster analysis from DNA microarray data have generated several commercially and publicly available tools for data visualization [1–5, 215]. However, such tools have the following drawbacks. Most of them do not implement dimensionality reduction to represent data on a 3D scatter plot, and the tools that do this functionality go no further than showing gene-points. On the other hand, these tools implement a small number of pre-fitted clustering methods, which raises problems when validating new methods. Moreover, they do not offer a visual interaction framework that is able to validate clustering results according to a reference partition from the data domain.

Based on the above discussion, we have designed several major features desirable in a tool of cluster analysis from DNA microarray data. These visualization components (features) are listed below with comments about their expected impact on the previously mentioned problems in cluster analysis.

1. DNA microarray data dendrogram analysis (MDA). This refers to the ability of the tested tools to incorporate dendrogram analysis, in the sense that the user can select and evaluate different cut-off levels in the dendrogram. This can help the users to find the most suitable number of clusters, and offers a global view of the constructed cluster hierarchy and the associated heat map for the available data.

2. Scatter plot analysis (SPA). This is the ability of the tool to implement some type of scatter plot analysis on microarray data. This kind of graph depends on

the level chosen in dendrogram analysis and can also be useful for validating the cluster number. Moreover, a scatter plot allows us to detect outliers or noise in the data, as well as to validate the inter-cluster distance used by the clustering methods.

3. Microarray parallel coordinates (MPC). Implementation of parallel coordinate analysis on the clusters represented as points and heat maps. With this kind of representation, a user can validate cluster homogeneity (cluster internal quality) by complementing the dendrogram and scatter plot analysis.

4. Microarray statistical analysis (MSA). Whether the tool integrates other statistical data analyses. This component can provide preparatory data analysis, such as feature, metric and clustering method selection.

5. Microarray data clustering methods (MCM). Whether the tool couples clustering methods. This component allows selection of a clustering method and compares the results with other methods. In this way, the method that best fits the data can be chosen, and thus includes the selection of data and inter-cluster distance implicitly.

From these points, Table 3.1 compares six tools from [1–5] and additionally, we have also included the 3D-VisualCluster tool (3D-VC), which was developed as part of this PhD Thesis, being the result of our proposed framework that will be explained later. This table presents visualization items as the rows and tool references as the columns. A check mark (✓) is used to indicate that a tool provides the specified property for that row, and "–" indicates that it does not provide this property. Additionally, the value "b-in" means that the tool implements the corresponding features (clustering methods or statistical measures) as built-in functionalities, and hence can not be extended without a considerable modification of the software. The value "ext" means the opposite of "b-in", since the tool has the ability of extending its functionality by linking with external software components (for example, through an R language API [16]).

Some of these visual components in Table 3.1 can be improved by extending their functionality, such as the case of Visual Exploration 3D ([5]) and the proposed 3D-VC, which both implement a 3D surface reconstruction. 3D-VC also improves the scatter plot analysis by implementing cluster boundary computation and a 3D reference

Table 3.1: Comparative table of visualization tools versus existing visualization components. Tools: *Cluster and TreeView of [1], TM4 of [2], GenCluster 2.0 of [3], HCE 3.5 of [4], Visual Exploration 3D of [5] and 3D-VisualCluster as 3D-VC.*

Features	Tools					
	[1]	[2]	[3]	[4]	[5]	3D-VC
DNA microarray data dendrogram analysis (MDA)	✓	✓	✓	✓(1)	–	✓
Scatter plot analysis (SPA)	✓(2)	✓(2)	–	✓	✓(4)	✓(2)(3)(4)(5)
Microarray parallel coordinates (MPC)	–	–	–	✓	✓	✓
Microarray statistical analysis (MSA)	b-in	b-in	b-in	b-in	b-in	ext
Microarray data clustering methods (MCM)	b-in	b-in	b-in	b-in	b-in	ext

(1) Dendrogram dynamic query control (DDQC). This lets users eliminate uninteresting clusters from a dendrogram and shows the interesting clusters more clearly [4].

(2) Microarray dimensionally reduction (MDR). The use of data dimensionality reduction techniques for microarray analysis on a scatter plot.

(3) Cluster boundary point (CBP). The ability of the tool to build the boundary of a cluster. That is, determining the cluster boundary genes, which allows the realization of further analysis without taking into consideration the cluster interior genes.

(4) 3D surface reconstruction (3D-SR). Visualization of clusters or other structures in the form of 3D surfaces as an alternative to the existing visualizations.

(5) 3D reference partition representation (3D-RPR). 3D visualization of the surfaces of a reference partition from the analyzed data set, and comparison of the reference partition with the clusters of the dendrogram.

partition representation. 3D-VC avoids implementation of clustering methods and statistical measures as built-in functionalities. Instead, it allows linking to other software components by the corresponding API, and hence is easily extendable to newly developed clustering methods or statistical measures through a general purpose language such as R.

Chapter 4

Evolutionary Algorithms

EVOLUTIONARY algorithms can be found within the *evolutionary computation* context, which is nothing less than an alternative approach to deal with complex search and learning problems through computational models of the evolutionary processes.

Evolutionary algorithms come to be precise methods of evolutionary computation, targeted at conducting some kind of stochastic search, in such a way that there exists interaction and repetitive selection of the most fit structures involved in the process.

Evolutionary algorithms with the best theoretical basis of the whole evolutionary computation are *genetic algorithms*. They are stochastic search algorithms based on the principles of natural selection and genetics. Genetic algorithms represent a powerful tool to solve complex optimization problems where the traditional methods face difficulty to find an optimum solution. Thus, they provide an efficient, adaptive and robust search and optimization processes that are usually applied to very large, complex and multidimensional search spaces.

A genetic algorithm is essentially comprised of a collection of individuals or chromosomes (called the population) and several biologically inspired operators that create a new (and potentially better) population from an old one, based on a selection process in favor of the most fit individuals (solutions). Each individual has an associated fitness value that indicates its degree of goodness with respect to the solution it represents. Several biological operators like selection, crossover and mutation are applied to the individuals of the population to yield more fit individuals.

This chapter introduces evolutionary algorithms as a via of resolution of complex optimization problems, focusing our attention on genetic algorithms. We first introduce the structure of an evolutionary algorithm and the aim of the optimization problems, which allow us to introduce, as the following step, genetic algorithms. In that section, the structure, basic procedures, pseudocode and discussion about some drawbacks of genetic algorithms are given. At the end of the chapter, it outlines a literature review about genetic algorithms applied to hierarchical clustering analysis, and their impact on the domain of DNA microarray data.

4.1 Evolutionary Computation

Darwinian evolution principle of “Survival of the most fit” captured the popular attention, mainly in the Engineering field. This principle is used as a starting point to introduce evolutionary computation (EC). Evolved biota demonstrates optimized complex behavior at each level: the cell, the organ, individuals and the population. Biological species have solved the problems of chaos, chance, nonlinear interactivities and temporality. These problems proved to be in equivalence with the classic methods of optimization. The evolutionary concept is applied to problems where heuristic solutions are not present or which leads to unsatisfactory results. As a result, evolutionary algorithms are of recent interest, particularly for solving practical problems.

Different organisms of an ecosystem can co-exist and compete for the same resources. The organisms that are most capable of acquiring resources and successfully procreating are the ones whose offsprings will tend to be numerous in the future. Less adapted organisms, for some reason, will tend to have few or no offsprings in the future [222]. The former are said to be more fit than the latter, and the distinguishing characteristics that caused the former to be fit are said to be *selected* for over the characteristics of the latter. Over time, the whole population of the ecosystem is said to evolve to organisms that, on average, are more fit than those of previous generations of the population due to they exhibit more of those characteristics that tend to promote survival.

These evolutionary principles are abstracted by EC techniques into algorithms that may be used to search for optimal solutions to a problem. In a search algorithm, a number of possible solutions to a problem are available and the task consists of finding the best solution possible in a fixed amount of time. An exhaustive search quickly becomes impractical as the search space grows in size. Traditional search algorithms randomly sample (e.g., random walk [228]) or heuristically sample (e.g., gradient descent [170]) the search space one solution at a time in the hopes of finding the optimum solution.

Thereby, we can say that the key aspect distinguishing an evolutionary search algorithm from such traditional algorithms is that it is population-based. Through the adaptation of successive generations of a large number of individuals, an evolutionary algorithm performs an efficient directed search. Evolutionary search is generally better than random search and is not susceptible to the hill-climbing behaviors of gradient-based search.

The rise of evolutionary computation has developed four historical paradigms that have served as the basis for much of the activity of the field: genetic algorithms [124], *genetic programming* [157, 158], *evolutionary strategies* [204], and *evolutionary programming* [182]. The basic differences between these paradigms lie in the nature of the representation schemes, the reproduction operators and selection methods.

4.2 Evolutionary Algorithm Structure

As previously mentioned, evolutionary algorithms (EAs) are stochastic search methods, inspired by the evolutionary principle, that is, survival of the most fit individuals (solutions) due to their adaptation to the environment. In other words, the most suitable individuals will have a better chance of surviving and therefore, a greatest possibility of passing their characteristics on to the next generations. To reach such evolution progress, an EA should have the following characteristics:

1. A population of possible solutions appropriately represented by individuals;

2. A selection process based on the fitness of the individuals;

3. A transformation process, which creates new solutions apart from the existing ones.

In the first point, the population of individuals (candidate solutions) reflects the beginning of a parallel stochastic search, differentiating itself from classic methods which carry out sequential search. An analysis on the effectiveness of EAs with respect to other methods can be found at [199, 222]. The second point refers to a selection process that favors to the best individuals of the current population, supporting to guide the search towards its goal. The last point assumes some kind of individual recombination process, so that the new individuals created out of such a transformation process have *valuable information* from their parents. Every iteration *transformation + selection* is called *generation*. The idea to follow for this reasoning is that, when the EA reaches certain number of generations, the best individual be close to the desired solution. According to the above, a general schema of an EA can be suggested in the following form:

1. Create initial population P_0;
2. Evaluate the fitness of individuals in P_0;
3. Repeat (a fixed number of generations g, with $i \in [1, g]$);
4. Select the most fit individuals of P_{i-1} to P_i;
5. Carry out the corresponding transformations on individuals in P_i;
6. Evaluate the fitness of individuals in P_i;
7. $i = i + 1$;
8. Check the stopping condition of step 3;
9. End.

4.3 Optimization Aim

In the following section, we study a type of evolutionary algorithm very commonly used in search and optimization problems, genetic algorithms, but before that, we examine some general matters regarding the optimization process important to focus us on genetic algorithms.

Optimization is the process of making something better. An engineer or scientist conjures a new idea up and optimization improves on that idea. Optimization consists of trying variations on an initial concept and using the information gained to improve on the idea. In general, when one needs to optimize a process or a function, the goals to be reached have to be very clear. A reflection upon this matter is presented by Beightler, Philips and Wilde in [105], that is:

> "Man's longing for perfection finds expression in the theory of optimization. It studies how to describe and attain what is Best, once one knows how to measure and alter what is Good or Bad... *Optimization Theory* encompasses the quantitative study of optima and methods for finding them".

This is the reason why optimization seeks to improve search performance towards optimum points, from which two aspects are deduced, the process of improvement of the search and the optimum point. Optimization methods commonly focus on convergence (optimum point) and do not pay attention to the performance of the method. In practice, the goodness of a solution to a certain problem is judged relative to competition with other found solutions, a fact that shows it is not judged by the best criterium. We can conclude this idea by saying that convergence towards the best solution is not the best way to go as our concern is to obtain better solutions with regard to others. Next, we have to keep in mind that the primary goal of optimization is improvement; attainment of the optimum is not the most important thing when we are dealing with a complex problem. Indeed, we can obtain some good, *satisficing* (introduced by Simon, 1969, see [242]) levels of performance. This way, the definition of the best solution is relative to the problem at hand, its method of solution, and the allowed tolerances. Thus the optimal solution depends on the person formulating the problem. Education, opinions, bribes, and amount of sleep are factors influencing the definition of the best. Some problems have exact answers or roots, and the best has a specific definition. Other problems have various minimum or maximum solutions known as optimum points or extremes, and the best may be a relative definition. According to that, finding the optimum in a complex system is impossible in most cases, however we can improve the solutions found to a given problem.

4.3.1 Multiple Objective Optimization

In many real world situations there may be several objectives that must be optimized simultaneously in order to solve certain problem. The main difficulty in considering multi-objective optimization is that there is no accepted definition of optimum, and therefore, it is difficult to compare one solution with another [27].

To solve multi-objective problems, all the objectives need to be treated together. In general, these problems admit multiple solutions, each of which is considered acceptable and equivalent when the relative importance of the objectives is unknown. The best solution is subjective and depends on the need of the designer or decision maker.

The multi-objective optimization is formally stated as follows [60]: Find the vector $\vec{x}^* = [x_1^*, x_2^*, \cdots, x_n^*]^t$ of decision variables which will satisfy the m inequality constraints:

$$g_i(\vec{x}) \geq 0, \;\; i \in [1, m], \tag{4.3.1}$$

the p equality constraints,

$$h_i(\vec{x}) = 0, \;\; i \in [1, p], \tag{4.3.2}$$

and optimizes the vector function,

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \cdots, f_k(\vec{x})]^t. \tag{4.3.3}$$

The constraints given in equations (4.3.1) and (4.3.2) define the feasible region $\mathcal{F}$ which contains all the admissible solutions. Any solution outside this region is inadmissible since it violates one or more constraints. The vector $\vec{x}^*$ denotes an optimal solution in $\mathcal{F}$. In the context of multi-objective optimization, the difficulty lies in the definition of optimality since we rarely find a situation where a single vector $\vec{x}^*$ represents the optimum solution to all the objective functions.

For a better understanding of all this, it is important to introduce to the domain of multi-objective optimization, the concept of *Pareto optimality* [118]. There are a set of optimal solutions, known as Pareto optimal ones, non-inferior solutions, or effective solutions. Without additional information, all these solutions are equally satisfactory. Then, the goal is to find as many of these solutions as possible. If reallocation of resources cannot improve one cost without raising another cost, then the solution is Pareto optimal.

A Pareto evolutionary algorithm returns a population with many members on the Pareto front (that is, the place where the optimum solutions are located). The population is ordered based on dominance. Solution $\vec{x}_1$ dominates solution $\vec{x}_2$, if $\vec{x}_1$ has a lower cost than $\vec{x}_2$ for at least one of the objective functions and is not worse with respect to the remaining objective functions. A solution is Pareto optimal if no other solution dominates that solution with respect to the cost functions. A solution is non-dominated if no solution can be found that dominates it. Once this set of solutions is found, the user can then choose a single solution based on various post-optimization trade-offs rather than weighting the costs.

A formal definition of Pareto optimality from the point of view of minimization problem may be given as follows [94]: A decision vector $\vec{x}^*$ is called Pareto optimal if and only if there is no $\vec{x}$ that dominates $\vec{x}^*$, i.e., there is no $\vec{x}$ such that:

$$\forall i \in [1,k], f_i(\vec{x}) \leq f_i(\vec{x}^*) \text{ and } \exists i \in [1,k], \text{where } f_i(\vec{x}) < f_i(\vec{x}^*).$$

A solution $\vec{x}^*$ strongly dominates a solution $\vec{x}$ if $\vec{x}^*$ is strictly better than $\vec{x}$ in all the objectives.

Thus, multi-objective optimization is interested in obtaining a set of non-dominated solutions. Hence, it is imperative that genetic diversity is maintained in the population. Several attempts pointing to this direction can be found in [69, 71, 106, 198, 224].

4.3.2 Combinatorial Optimization

An important part of the optimization problems are the combinatorial optimization problems since they play a major role in the resolution of practical problems (in our case, the clustering problem) from *discrete mathematics*. The combinatorial optimization contains a huge body of problems with different features and properties. Although these problems are quite different from each other, the problems can be characterized as follows:

- Determining a permutation of some items associated with the problem.
- Determining a combination of some items.

- Determine both permutation and combination of some items.
- Any one of the above subject to constraints.

A common feature of such problems is that if the permutation and/or combination can be determined, a solution can then easily be derived with a problem-specific procedure. So the general approach for applying genetic algorithms to these problems is to obtain a appropriate permutation or combination of items under consideration. To then use a heuristic to construct a solution according to the permutation and combination.

4.4 Genetic Algorithms

Genetic algorithms (GAs) occupy the central space within evolutionary algorithms (EAs), being the most complete paradigm of whole evolutionary computation for both reasons, practical as well as theoretical ones.

GAs can be viewed as blind search methods, inspired by the natural selection mechanism; their goal is to randomly combine the most fit string structures (survivors) to form new strings, so that as this process is repeated many times, the new individuals created from previous generations, will be better and better. While randomized, GAs are no simple random walk. They efficiently exploit valuable information in string structures to speculate over new points in the search space, providing a desired improvement in global search performance.

GAs were first presented by John Holland and his colleagues in 1975 [124], at the University of Michigan. The aim of their creation was:

- To give an abstract and rigorous explanation of *adaptive* processes of natural systems;
- To design an artificial system tool that *preserves* the important mechanisms of natural systems.

This approach has led to important discoveries in both natural and artificial systems science. Ever since their creation, researchers have focused on the robustness of GAs, that is, the balance between *efficiency* and *efficacy* to find new solutions, which is necessary for survival in any environment. Costly redesigns of artificial systems can be reduced or removed if these are made more robust. Moreover, artificial systems with a high level of adaptation will perform their functions longer and better. Features as self-repair, self-guidance and mating are the rule in biological systems, whereas they barely begin to be present in complex artificial systems. A reasoning over the previous point can be found at [105]; yet, we will emphasize some of the most important characteristics that make GAs robust, that is:

- GAs constitute the most complete approach of evolutionary computation (EC), that is, they gather in a natural way, all the fundamental ideas of such an approach;
- They are very flexible, that is, they can easily adopt new, general or specific ideas that emerge in the field of EC. Moreover, they can easily hybrid with other paradigms or approaches, even though they have nothing to do with EC;
- GAs are the paradigm with the greatest theoretical basis among the ones of EC. What is more, such a theoretical basis is simple in its development, offering great possibilities of expansion;
- Between all other paradigms of EC, they are the ones that require less specific knowledge for their performance, and therefore, the most versatile ones. Furthermore they can incorporate specific knowledge with little additional effort;
- They are easily implemented in medium capacity computers, giving rather acceptable results as far as precision and employed resources are concerned for a great quantity of problems handled with difficulty by other methods;
- As a consequence to the previous arguments, GAs possibly are by far the most used paradigm among the ones of EC and, along with *neural networks*, the most used in whole *natural computation*. This is important as it provides users with a great variety of empirical trials offering, operators, parameters and specific implementations for a wide variety of problems.

A accepted definition of GAs, which has been commonly used can be found in [199], that is:

Genetic Algorithms are stochastic methods of blind search of quasi-optimal solutions. They have a population that represents a set of possible solutions, which is subjected to certain transformations through which, it is intended to obtain new candidates, and to a selection process slanting in favors of the best candidates.

This definition states the principal characteristic of GAs, namely, in the search method domain, GAs incorporate a mechanism of *selection* of solutions, which consists of two parts:

1. Selection of the elements where the transformation will be applied (*selection operator*), and
2. Selection of the elements that will survive (*replacement operator*).

These characteristics along with encoding, multiple and stochastic blind search are introduced to GAs, giving them greater search robustness. We now should clarify some previous terms, as:

- *Blind search*, GAs have no kind of specific knowledge of the problem, so the search is based exclusively on the values of the objective function;
- *Encoding*, they do not work directly on the domain of the problem, but on representations of their elements;
- *Multiple search*, they simultaneously search within a set of candidate solutions and,
- *Stochastic search* is referred to the selection phase as well as to the transformation one. This provides control over the penetration factor of the search.

Last but not least, a characteristic that should not be left unstressed is the implicit parallelism of GAs, which means that, from an external point of view, they process code strings, whereas internally, they do much more than that. When GAs are processing every string of a population, they indeed are processing all similarity patrons that those strings contain. See *the schema theorem* [124].

4.4.1 Genetic Algorithm Structure

As mentioned earlier, a GA transforms a set of candidate solutions of certain problem to update the search incorporating as well, a selection process that favors the most fit individuals. It is expected that, within a few generations, the best individual represents a candidate, close enough to the solution we are looking for.

GAs as part of evolutionary algorithms (EAs), present the same schema of EAs, even with some special features. A general structure of a GA can be seen in the diagram of Figure 4.1.

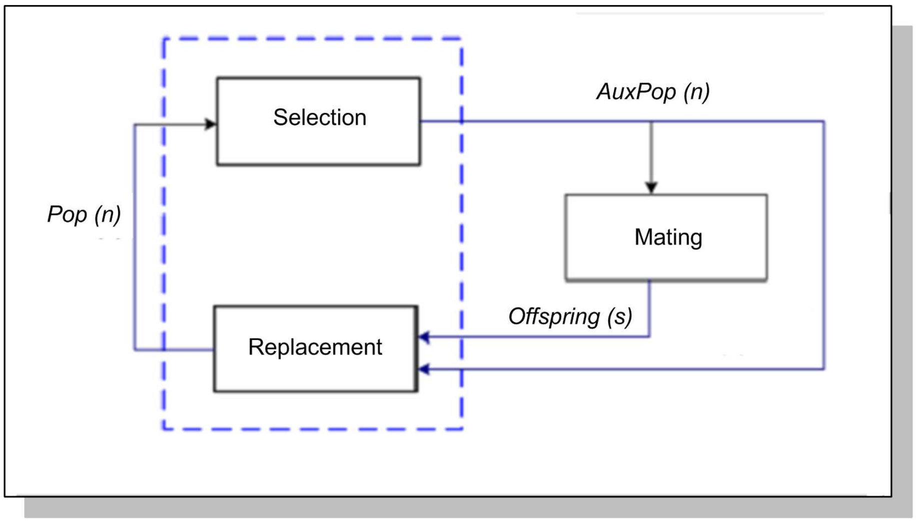


Figure 4.1: Basic loop of a genetic algorithm.

As observed in this figure, a population of n members (*Pop*) is subjected to a selection process to create an intermediate population *AuxPop* of n breeders. From that population it extracts a reduced group of individuals called *progenitors*, being the ones that finally *mate*. After applying genetic operators, s individuals are obtained to form the offspring. In the *replacement* phase, n survivors have to be selected from $n+s$ individuals comprising the auxiliary population and the offspring to form the new population.

The objects participating in the GA evolution process are usually designed as strings v (also called, *chromosomes*), representing an encoding of elements x in the search space. Both elements receive the name of individuals. Every chromosome v consists of m positions that represent various attributes (called also genes), which are generally codified in $l = L_1 + \cdots + L_m$ bits (or any other measure of *quantity of information*), where L_i is the number of bits needed to represent the attribute of position i. The structure of an individual is usually called genotype, and its content, phenotype. Under an implementation, the genotype is the class of objects and the phenotype is formed by objects (class instances) of the class.

4.4.2 Criteriums and Mechanisms used in GAs

Before showing the basic procedures of a GA, it is necessary to keep in mind some criteriums and mechanisms that should not be absent in the use of GAs, namely:

1. *Encoding criterium:* we have to specify the process (encoding) which makes every point of the problem domain corresponds to a representation, so the mechanism of passing from genotype to phenotype;

2. *Criterium of treating of non-feasible individuals:* it is not always possible to establish a point-by-point correspondence between the problem domain and the set of encodings used to solve the problem, that is, search space encoding. Consequently, not all strings encode valid elements of the search space, therefore useful procedures should be built to distinguish them;

3. *Initialization criterium:* refers to how an initial population that starts the basic loop of the GA should be constructed;

4. *Stopping criterium:* we have to set the conditions under which it is considered that the GA has reached an acceptable solution or otherwise, the search has been a failure and there is no point in continuing;

5. *Fitness function:* the most suitable fitness function for the problem has to be set;

6. *Genetic operators:* are called the operators with whom mating is carried out. All GA use at least two genetic operators, crossover and mutation; however, they are not the only possible ones and variations of them are also accepted;

7. *Selection criterium:* selection has to guide the search process in favor of the most fit members but there are many ways to succeed this;

8. *Replacement criterium:* the criterium with which breeders are selected do not necessarily have to be the same through which survivors are selected, thus they need to be specified separately;

9. *Performance parameters:* a GA needs to be given certain performance parameters, such as population size, application probability of the genetic operators, encoding precision, convergence tolerance, etc.

As a mechanism of population sampling, that is, selection of a subset of individuals from a population, we have the following samplings:

1. *Direct sampling:* we take out a subset of individuals in a population, following a fixed criterium such as "the k best" or "the k worst", etc.

2. *Simple or equiprobable random sampling:* the same probability to form part of the sample is given to all elements of the base population and the sample is built through a simple Bernuolli trial;

3. *Stochastic sampling:* in this sampling, probabilities of selection or scorings are given to the elements of the base population on the basis of their fitness. By default, scoring p_i associated with individual x_i of population $\mathcal{P} = \{x_1, \ldots, x_n\}$ is calculated as the relative fitness of such individual, being $u_1, \ldots, u_n$ respectively, and holding the following formula:

   $$p_i := \frac{u_i}{u_1 + \cdots + u_n}, \forall i \in [1, n],$$

   guaranteeing $p_1 + \cdots + p_n = 1$.

   There are many mechanisms of stochastic sampling, but here we will only explain sampling by drawing (or roulette wheel method), which is the most used. This is shown below:

   (a) accumulated scorings are calculated
   $q_0 := 0$
   $q_i := p_1 + \cdots + p_i, \forall i \in [1, n]$

(b) simple random numbers k are generated, that is,
$r_j \leftarrow$URand[0, 1), $\forall j \in [1, k]$

(c) $\forall j \in [1, k]$ individual x_i is selected, which verifies:
$q_{i-1} < r_j < q_i$.

4.4.3 Basic Procedures of a Genetic Algorithm

A GA consists of three basic procedures called selection (of breeders), mating (transformation) and replacement (or survivor selection), which are explained below.

- *Selection process:* consists of sampling from the initial population, n members of the breeder population.

- *Mating process:* alteration and transformation operators are applied (mutation, crossing, inversion, etc.) over certain members of the breeder population (intermediate population) and an offspring of s new members is obtained. Different combinations of genetic operators can be found at [73, 85, 89].

  The most used genetic operators are:

  The *crossover operator* (recombination or exploitation operator) normally acts over two individuals (named parents) to originate two other individuals (named children) that combine characteristics of the parents. Since the individuals are represented as strings, the crossover is carried out through an interchange of segments, as shown in Figure 4.2.

  The *mutation operator* (alteration or exploration operator) is a unitary operator, carries out a small alteration in some of the genes of some individuals. Generally, the mutation operators are in charge of carrying out an in-depth search, that is, they provide a guarantee of access to all points in the search space, while the crossover carries out a wide search, exploring new areas in the space, thanks to its interchange of information. Note that the mutation operator is able to retrieve information that could lost in the recombination process of the crossover operator. Figure 4.3 shows a mono-point mutation of a string.

- *Replacement process:* starting from n individuals of the initial population plus s individuals of the intermediate population (offspring), a new population of n individuals has to be retrieved.

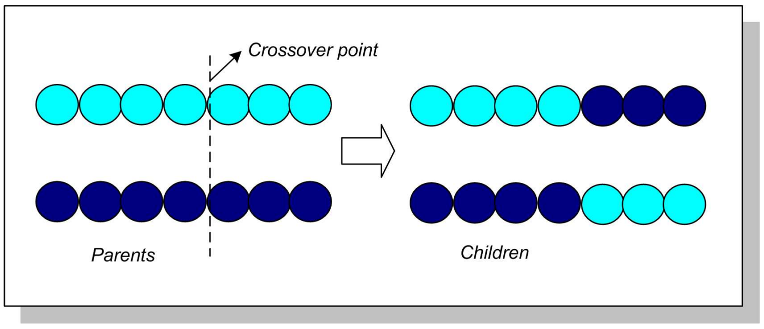


Figure 4.2: Crossover of two chromosomes in one single point.

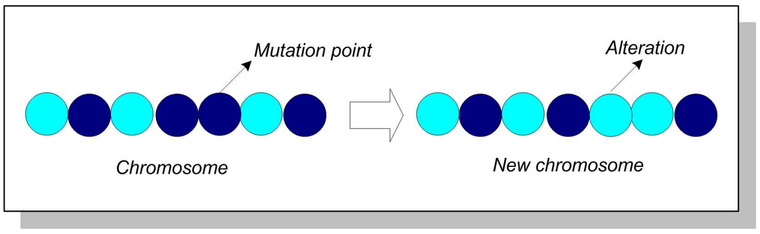


Figure 4.3: Mutation of a chromosome in one single point.

There are several replacement criteriums as it can be verified in the literature:

1. *Immediate replacement:* the s offsprings substitute respectively their parents;
2. *Replacement by filling factor:* the s offsprings substitute those members of the breeder population that are most similar to them;
3. *Replacement by insertion:* according to the relative size of the offspring with respect to the population, two cases are distinguished:
   – if $s \leq n$, then s members in the population of breeders are sampled to be removed (under certain criterium, usually the worst). Those members are replaced by the offsprings.

  - if $s > n$, then n members of the population of offsprings are sampled and the new population is built with them. Note that in this way, any individual can only live at most one generation.

4. *Replacement by inclusion:* both s offsprings and n parents are joined in a single population where n members are sampled (usually, the best ones).

In the literature, the authors sustain that a good criterium for a GA is to incorporate selection through universal stochastic sampling (roulette wheel method) in the conservative category; replacement is usually either immediate or by insertion (with the same sampling).

Having considered everything up to now, we now can give the pseudocode of GA through of Algorithm 4.4.4 listed below.

4.4.4 Pseudocode of a Genetic Algorithm

Output: P, a population representing the individuals of the last generation evolved from the initial population.

```
$t := 0$; % start with an initial time.
InitPopulation $P(t)$; % initiate the population of individuals (usually random).
EvalFitness $P(t)$; % evaluate the "fitness" of the individuals.
% Assume condition$(t)$ as a formula of the propositional calculus.
while not condition$(t)$ do % check stopping criterium (time, fitness, etc.).
  $t := t + 1$; % increase timer.
  $P' :=$ SelectParents $P(t)$; % select a sub-population for the mating process.
  $P'' :=$ Crossover $P'(t)$; % recombine genes of the randomly selected parents.
  $P''' :=$ Mutation $P''(t)$; % modify genes of the randomly selected chromosomes.
  EvalFitness $P'''(t)$; % evaluate their new fitness.
  $P :=$ Survivors $P, P'''(t)$; % select survivors.
endwhile
```

end GA.

A *simple genetic algorithm* particularly incorporates the following methods and criteriums [229]:

1. *Encoding criterium:* specific for every problem. It has to make every point of the problem domain correspond to an element of the search space, which necessarily consists of unsigned binary integers, that is, simple binary strings.

2. *Treating criterium of non-feasible individuals:* it does not exist. It is considered that codification is carried out in such a way that all possible strings represent feasible individuals.

3. *Initialization criterium:* the initial population consists of randomly chosen binary strings.

4. *Evaluation and fitness functions:* the fitness function coincides with the evaluation one. Preferably, the evaluation function is given through an objective function.

5. *Genetic operators:* mono-point crossover and mutation bit by bit over codified individuals.

6. *Selection criterium:* by drawing lots.

7. *Replacement criterium:* immediate.

8. *Stopping criterium:* fixing maximum number of iterations.

9. *Performance parameters:* discretionary. It depends on the problem to deal with.

Up to this moment, we have presented the principal characteristics and the performance principle of GAs. For a formal presentation (*theoretical fundamentals*) that this approach works at least on binary strings, consult [105], [199], [186], [185], where a proof using the *schema* concept as well as an in-depth study will be found.

The traditional proof of convergence for the binary GA is called the *schema theorem* [124]. A schema is a string of characters consisting of the binary digits 1 and 0, and an additional "don't care" character, $*$. Thus, schema $11{*}{*}00$ means the center two digits can be either a 1 or a 0 and represents the four strings given by 110000, 110100, 111000, and 111100. The schema theorem says [105]:

Short schema with better than average costs occur exponentially more frequently in the next generation. Schemas with costs worse than average occur less frequently in the next generation.

The idea is that the most fit schemas survive to future generations. Following the best schema throughout the life of the GA provides an estimate of the convergence rate (not necessarily to the global minimum) to the best chromosome. Thus, the ability of the algorithm to explore and exploit simultaneously, a growing amount of theoretical justification, and successful application to real-world problems strengthens the conclusion that GAs are a powerful, robust optimization technique.

4.5 Difficulties in the use of Genetic Algorithms

Although the GAs robustness with respect to other methods is verified, the problem of finding genetic operators and an adequate encoding is still the greatest difficulty. Moreover, the issues of solving a problem with acceptable quality through a GA and building an appropriate fitness function correlated with the candidate solutions are also of vital importance. In general, fitness functions usually are expensive (implicitly there is encoding and decoding of individuals), hence, approximations of them to minimize their calculation complexity usually are effective.

We should also point other problems present in GAs that have to be treated if we want to achieve good results throughout the process. A typical problem concerning GAs is the lack of *diversity*, a factor that has to be controlled in every moment, as its immediate consequence is *premature convergence* towards local optimums. A possible solution to this problem is to run the GA in two stages, in the first stage, make sure that the GA carries out an in-depth search, which means to increase the mutation probability without guiding the search (no selection, ensuring diversity on individuals) and in the second stage, to run the GA in its normal form.

Some tips to follow towards a good use of GAs are:

1. Never draw conclusions from a single execution:

- use statistic measures (arithmetic means, medians, etc.);
- with an enough number of independent executions.

2. "We can obtain whatever we want in an experimentation according to the difficulty of the used cases". We should not adjust/check the algorithm performance over simple examples if we want to work on real cases.

3. From the application point of view, double approach and different design:

    - finding a very good solution at least once;
    - at least, finding a very good solution in every execution.

4.6 Genetic Algorithms and Hierarchical Clustering

In this chapter, we have stated by way of introduction, the most important characteristics of GAs as well as the principles on which they are based. We now are going to highlight their application to our area of interest, the hierarchical clustering problem for gene expression data, and discuss existing approaches for this purpose.

The GAs application to *data mining* is being of great importance in the knowledge discovery processes from the study of classification problems, where an optimization criterium is required [113,190,192]. This way, GAs applied to clustering problems [72, 129,173,257,258] are very useful in *DNA microarray data analysis* [35,49,100,137,227], providing understanding of the processes related to gene functions. Consequently, those mining processes are of vital importance for the human health research.

According to the above, there has been recently an increasing interest in dealing with the clustering problem using genetic algorithms, principally for the partitional approach [51, 59, 78, 92, 164, 180]. GAs have intensively been applied to the partition problem of a data set, namely, the distribution problem of n objects in M sets according to some minimization criterium of an additive optimization strategy on those sets [109]. Once the optimization criterium is chosen, the clustering problem consists of finding an efficient algorithm able to carry out a search on the classification space. However, the clustering partitional problem is much less complex than the hierarchical clustering one, where the search space is extremely large.

GAs approaches building hierarchical clustering are a topic that is almost untouched in the literature. According to [129], this is probably due to the fact that it is not straightforward to define a fitness function capable of guiding the evolution of dendrograms. Indeed, few works exist that attempt to apply evolutionary techniques to build hierarchical clusterings. In fact, after reviewing the literature only three works that deal with the previous approach have been located as explained below.

In [169], a genetic algorithm is used for searching an optimum hierarchical clustering on a data set according to the least square method. Through a different approach to our, this work is based on the ultrametric dissimilarity notion, which is used to find the best hierarchical clustering (the best, theoretically) on a data set since there is a bijection between hierarchical classifications and the set of ultrametric dissimilarities. Therefore, they deal with the problem of searching directly in the set of ultrametric dissimilarities rather than searching in the hierarchical classification space. In addition, the authors of [169] conclude that by finding the ultrametric dissimilarity closest to the used data set, we would find the best hierarchical classification. Finally, the experiments were carried out on very small non-biological data sets.

In [108], a hierarchical, incremental, and unsupervised clustering algorithm was defined. This algorithm is based on the work performed in [109], and it is distance-based. Since it computes centroids for each cluster and then, builds a hierarchy of subsets: a partitioning subset may itself be partitioned and those subsubsets may themselves be partitioned and so on. For this reason, the previous algorithm is related to K-means method [82]. In order to fulfill the previous requirements, this algorithm is combined with the power of the GAs evolutionary force. Nevertheless, data and the proximity matrix were represented on a nominal scale and as a consequence, the *Hamming distance* was used. On one hand, this algorithm is not explicitly oriented to the search of a global optimum. On the other hand, it was applied on non-biological small data sets.

The approach of [169] is very different to our, in the sense that, the search space of each method is different and so, their objective functions are different, too. Furthermore, the computational complexity of this algorithm grows exponentially for large data sets, as the case of DNA microarray data. The method of [108] has the drawback of having a nominal scale (non-real values) on both, data set and proximity matrix, which restricts its use. Moreover, the approach of centroid computation for hierarchical clustering can imply a time overload, and is biased toward the discovery of spherical clusters, which clearly is inappropriate for many applications [129]. Additionally, both works [169] and [108] carried out for other data contexts very different of gene expression data, and neither of them, introduce heuristics to reduce the complexity of the search space. According to latter, the first experiments carried out on our clustering evolutionary model without including constraints to the search space (as made in [169] and [108]), showed our approach intractable according to runtime when the data set is large. This suggests that the GA-based search of an optimum dendrogram without including constraints still remains intractable, in general. Hence, the approaches of [169] and [108] would also be intractable for large data sets.

Recently in [43], a GA for finding cluster hierarchies (called GACH) was defined. This algorithm is based on the combination of genetic algorithms, information theory and model-based clustering. Therefore, the individuals of GACH represent hierarchies of clusters on a data set, and the applied fitness function relates the clustering problem with that of data compression by Huffman coding. Particularly, this method is applied to the field of medical image processing and not to the field of DNA microarray data. On the other hand, this one builds cluster hierarchies that are not dendrograms, which makes the analysis and the comparison (statistical as well as visual) with other hierarchical clustering methods be almost impossible. Moreover, the dendrograms are a powerful tool in biological data analysis (and visualization), and there are many concepts built on this base [8]. Thus, in this research, we only consider approaches representing cluster hierarchies being dendrograms. Consequently, the two previous features of GACH make it out of our scope.

Chapter 5

Complexity of the Research Hypothesis

This chapter presents the first contribution of this research work by classifying the problem addressed in our research hypothesis (see Chapter 1) in class NP-complete. This class contains the most difficult problems to solve of all kinds of hard problems. Starting then from the hypothesis given in Chapter 1, we have made an analysis of the algorithmic complexity of the problem, allowing us to characterize the search space. Knowing the particularities of the search space on which, we are going to work, provides knowledge which is useful to develop heuristics and constraints to the problem, making more efficient the search process towards a global optimum. To this end, we have first introduced the fundamental theorem of this research, formalizing in that way, the hypothesis. After that, we have introduced the basic concepts of the NP-completeness theory, which support the proof of the fundamental theorem, while also providing additional results from the search space. The proof of the fundamental theorem and the remaining results in this chapter are the basis of the clustering evolutionary model that has been introduced in the next chapter in order to face the problem given by our research hypothesis.

5.1 The Problem

This section formalizes the first part of our research hypothesis (Chapter 1) which states that *the problem of finding the best dendrogram on a data set, that is, the problem of finding the optimum dendrogram from a data set according to an objective function measuring the dendrogram quality is an NP-complete problem.* This conjecture can be rewritten in theorem form, specifying formally the details, namely:

Theorem 5.1. *Fundamental theorem of this research.*
Given a data set $\mathfrak{P}_n$ of size n, f and f' two objective functions for a clustering and a dendrogram of $\mathfrak{P}_n$ respectively, then *the Problem of Finding the Best Dendrogram on* $\mathfrak{P}_n$ (denoted as PFBD) *is an NP-complete problem.*

As will be seen, this theorem is the basis of our research for both theoretical and practical reasons. Because of that, the used evolutionary model that attempts to solve this problem is justified as an alternative technique to the existing methods. On the other hand, f' assumed as a well defined function on a dendrogram $\mathfrak{G}$, should necessarily include in some way, fitness (or contributions) f of clusterings $\mathfrak{C}_i$ in $\mathfrak{G}$, and without losing generality, we can also assume $f'(\mathfrak{G}) = \mathrm{OP}_i^n[f(\mathfrak{C}_i)]$. Where OP is one of the operators $\sum, \prod$, max, etc., or the combination of them, which are able to summarize contributions f of clusterings in $\mathfrak{G}$.

Before showing a proof of this theorem, it is necessary to enunciate some definitions and properties of the NP-completeness theory. Henceforth, we give concepts that will help to understand the proof of the fundamental theorem.

5.2 NP-Completeness

There exists a large family of computational problems for which fast algorithms have never been found, but neither have such algorithms been proved to be unattainable [24, 140]. This family is not just a list of seemingly difficult computational problems. It is in fact bound together by strong structural ties. The collection of problems, called the NP-complete problems, includes many well known and important questions in mathematics. NP-completeness tells us that they are all, in a precise sense, equally hard. On the other hand, if any one of these hard problems has a polynomial time algorithm, then they all do.

The question of the existence or non-existence of polynomial time algorithms for the NP-complete problems rates as the principal unsolved problem that faces theoretical computer science today [19, 97, 247]. Hence, the main characteristics of these problems can be listed as follows:

1. All these problems seem to be computationally very difficult, and no polynomial time algorithms have been found for any of them;
2. It has not been proved that polynomial time algorithms for these problems do not exist;
3. But this is not just a random list of hard problems. If a fast algorithm may be found for one NP-complete problem, then there also are fast algorithms for all of them;
4. Conversely, if it could be proved that no fast algorithm exists for one of the NP-complete problems, then there could not be a fast algorithm for any other of those problems.

Our next task is to develop an item machinery that permits to give precise definitions of all concepts that are needed:

- *Decision problem*: a decision problem is one that asks only for a yes-or-no answer. Many of the NP-complete problems can be formulated as decision problems or as optimization problems. Usually, if it finds a polynomial algorithm for a decision problem then with just a little more work, the corresponding optimization problem will be solved. Hence, it is possible to restrict the optimization discussion to a decision problem.
- *Language*: since every decision problem has two answers "Y/N", we can transform a decision problem as asking if a given word (the input string) is or is not a one of certain language. The language is the set of all words for which the answer is "Y".
- *Class P*: a decision problem belongs to class P, if there exits an algorithm A and a number c such that for every instance I of the problem, the algorithm A will produce a solution in time $O(n^c)$. n is the number of bits in the input string that represents I. Simpler, P is the set of easy decision problems.

- *Class NP*: a decision problem Q belongs to NP if there exits an algorithm A that holds the following:

    1. there is a certificate $C(I)$ associated to each word of the language Q (i.e., with each instance I for which the answer is "Y") such that, if the pair $(I, C(I))$ are input to algorithm A, then it recognizes that I belongs to the language Q.
    2. if I is a word that does not belong to the language Q, then there is no choice of certificate $C(I)$, in such a way that A recognizes I as a member of Q.
    3. algorithm A operates in polynomial time.

  In other words, NP is the class of decision problems, for which it is easy to check the correctness of a claimed answer, with help of a little extra information. So, it is not asking for a way to find a solution, but only to verify that the given solution is indeed correct.

  Summing up about both classes, in P are the problems where finding a solution is easy, and in NP are the problems where it is easy to check a solution that may have been very tedious to find.

- *Reducibility*: let Q and Q' be two decision problems. We say that Q' is quickly reducible to Q, if each instance I' of the problem Q' can convert it, with only a polynomial effort amount, into an instance I of Q. It holds that I' and I have the same answer ("Yes" or "No") for the same input. Namely, a computer program to solve Q, can be used to solve Q', with just a small amount of extra work.

- *NP-completeness*: a decision problem is NP-complete, if it belongs to NP and every problem in NP is reducible to it in polynomial time.

On the reducibility and NP-completeness concepts arise some implications very important for the proof of the previous theorem, that is:

1. If there exists a polynomial time algorithm for previous problem Q', then there exists one for Q;

2. A problem Q is NP-hard, if every problem in NP has a polynomial time reduction to Q. If, in addition, Q is in NP, then it is NP-complete;

3. Thus, *if Q' is NP-complete, and it has a reduction to another problem Q in NP, then Q is also NP-complete.*

For all the above, the NP-complete problems are considered the hardest ones in NP [153, 193, 194], and we are now able to provide a proof of the fundamental theorem.

5.3 Proof of the Fundamental Theorem

From the previous section deduces that to prove Theorem 5.1, we must first prove that PFBD is in NP and after, that it is NP-complete. But before, we have to do certain transformations and give some important propositions of the problem.

Let us begin, giving a representation of the search space (the dendrogram space) as a weighted graph, that is, a type of tree, Figure 5.1. The vertices (nodes) denote clusterings and the edges denote nested clusterings for a dendrogram in this space.

This tree representation is an agglomerative one of the search space, starting from root node $C_{1,1}$ to end node $C_{n,1}$, but a divisive way can also achieved in reverse sense. In this tree, there are n levels (levels of a dendrogram of $\mathfrak{P}_n$) and every possible clustering (vertex) used to build a dendrogram is located in one of these levels. Thus, vertex $C_{1,1}$ represents the only one clustering of level 1, where each data object of $\mathfrak{P}_n$ forms a cluster. Note that notation $C_{i,j}$ means, clustering j belongs to level i of a dendrogram on $\mathfrak{P}_n$.

From level 1, we can only generate k_2 different clusterings (Figure 5.1) by merging two clusters of $C_{1,1}$. This way, level 2 is reached. The same process from level 1 is applied to level 2 to obtain level 3, and so on. k_i is the number of clusterings generated by each clustering of level $i-1$ where i is the current level. In other words, k_i is the number of child nodes generated by each father node of the above level. Therefore, the number of nodes in a level i is obtained of the result of multiplying corresponding k_i with all k_j where $j < i$. Then, every level of the tree in Figure 5.1 is built from all possible clusterings formed by merging two clusters of clusterings in the above level. Note that moving from one level to the next lower one on this tree, the number of nodes increase at a rate of n^2, which is translated into several mathematical propositions shown below.

Considering that the tree in Figure 5.1 represents a graph $\mathfrak{U} =< V, E, W >$, where V is the set of vertices, E the set of edges and W the weight matrix in this graph, then the following propositions hold.

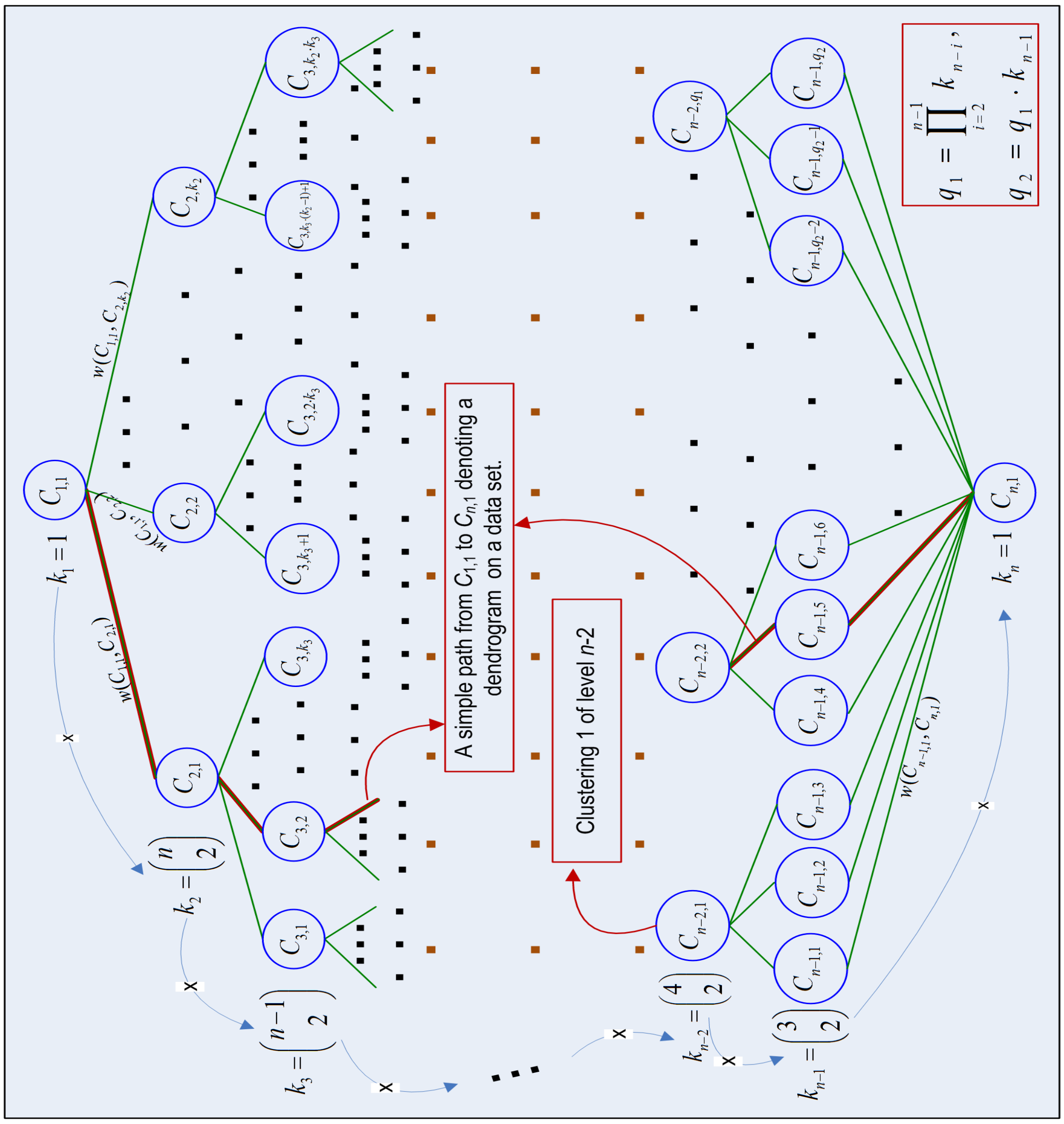


Figure 5.1: A representation of the dendrogram space in form of tree (a weighted graph). Every vertex denotes a clustering, the edges are connections between clusterings nested within the same dendrogram and w is a weight function. A simple path from node $C_{1,1}$ to $C_{n,1}$ means a dendrogram of the search space.

Proposition 5.1. *On the number of vertices and edges.*
Let $\mathfrak{U} =< V, E, W >$ be a graph on data set $\mathfrak{P}_n$, satisfying the conditions in Figure 5.1, then the number of both vertices $|V|$ and edges $|E|$ is given in the following expressions:

$$|V| = \sum_{i=2}^{n-1} \prod_{j=2}^{i} \binom{n-j+2}{2} + 2; \tag{5.3.1}$$

$$|E| = |V| + \prod_{i=2}^{n-1} \binom{n-j+2}{2} - 2. \tag{5.3.2}$$

Proof: This proof is obtained by inspecting the number of nodes that each level in the tree in Figure 5.1 has. Note that only one edge reaches each node (except for $C_{1,1}$ and $C_{n,1}$) and from each node emerges k_i edges to nodes of the next lower level. $\square$

Proposition 5.2. *The best path.*
Given a graph $\mathfrak{U} =< V, E, W >$ from data set $\mathfrak{P}_n$ and satisfying the conditions in Figure 5.1, then the algorithmic complexity to find the best path (the optimum path) from $C_{1,1}$ to $C_{n,1}$ is $O(n^{2n})$.

Proof: Expanding the sum and product in the expression of $|V|$ in Proposition 5.1, afterward, taking the maximum term out, we then obtain expression $\left(\frac{n^2}{2}\right)^{n-2} < n^{2(n-2)} < n^{2n}$ (for $n > 2$), which proves the result. Note one can assume that $w(C_{i,j}, C_{i+1,j'}) = f(C_{i+1,j'})$. Namely, weight function w is interpreted as contribution f of moving from a clustering to another in a dendrogram. $\square$

5.3.1 Transforming PFBD and Proving its NP-Completeness

As previously explained, an optimization problem as PFBD can be expressed as a decision problem. If without loss of generality, PFBD is assumed as a minimum problem, then the following statements are equivalents, meaning the same problem PFBD:

Finding the best dendrogram on a data set.

$\Updownarrow$

Given a K, Is there a dendrogram $\mathfrak{G}$ in $\mathfrak{U}$, $f'(\mathfrak{G}) \leq K$?

$\Updownarrow$

Given a K, Is there a path in $\mathfrak{U}$ from $C_{1,1}$ to $C_{n,1}$ less than or equal to K?

This way, we have PFBD in form of a decision problem and besides, it is easy to verify the correctness of a given solution to a PFBD instance, since it consists of adding (summarizing) all contributions f of the clusterings in the found solution-dendrogram together to obtain the final evaluation from f' (see statement of Theorem 5.1). Moreover, since PFBD is $O(n^{2n})$, it then *is in NP*.

We now only need to prove that PFBD is an NP-complete problem. For this, it is enough to reduce a problem of the NP-complete class to PFBD. Consider the following problem definition:

Definition 5.1. *Boolean Satisfiability (SAT Problem).*
Given a boolean formula φ in rCNF form over variables $u_1, \ldots, u_m$. That is, $\varphi = \xi_1 \wedge \xi_2 \wedge \ldots \wedge \xi_n$, where ξ_i are disjunction clauses ($\vee$) of r literals u_j (or their negations $\overline{u}_j$, $3 \leq r \leq m$) as a maximum. If $z \in \{0,1\}^m$, then $\varphi(z)$ denotes the value of φ when values z are assigned to the variables of it (where we identify 1 with True and 0 with False). Therefore, a formula φ is *satisfiable* if there exists some assignment z such that $\varphi(z)$ is True. Then, the *SAT problem* consists of finding a z such that $\varphi(z)$ be satisfiable.

SAT was the first problem classified as NP-complete [19, 64, 65, 97], being considered as the canonical problem for which the other NP-complete problems can be reduced to it. At this point, we can reduce SAT to PFBD through a suitable transformation:

Let φ be a rCNF ($r \geq 3$) formula on m variables with n clauses. We define a graph $\mathfrak{U}$ of $|V|$ vertices as defined in Figure 5.1 in the following way: the set of q_i vertices (q_i defined as in Figure 5.1) of level i in $\mathfrak{U}$ is associated to clause ξ_i of φ. The vertices in the set associated with a clause ξ_i correspond to q_i possible partial assignments of r variables ξ_i depends on (we call *partial assignments*, since they only give values for some of the variables [23]). If ξ_i depends on less than q_i vertices then we repeat one of the partial assignments. An edge between two vertices of consecutive levels in $\mathfrak{U}$ is put, if they correspond to consistent partial assignments and their partial assignments evaluate True, for these associated clauses in φ. Two partial assignments are *consistent* if they give the same value to all the variables they share. In addition, we put edges between every remaining two vertices of consecutive levels in $\mathfrak{U}$ following Figure 5.1.

As assumed that $w(C_{i,j}, C_{i+1,j'}) = f(C_{i+1,j'})$, we now make the following weight updating on $\mathfrak{U}$:

$$f(C_{i,j}) = f(C_{i,j}) + \xi_i(\vec{x}_j); \tag{5.3.3}$$

$$w(C_{i,j}, C_{i+1,j'}) = w(C_{i,j}, C_{i+1,j'}) + [\xi_i(\vec{x}_j) \wedge \xi_{i+1}(\vec{x}_{j'})]. \tag{5.3.4}$$

$\vec{x}_j$ and $\vec{x}_{j'}$ represent binary partial assignment vectors of literals u_i in clauses ξ_i and ξ_{i+1} respectively. $\vec{x}_j$ and $\vec{x}_{j'}$ are the partial assignments associated to vertices $C_{i,j}$ and $C_{i+1,j'}$ in $\mathfrak{U}$ respectively.

We claim that φ is satisfiable if and only if $\mathfrak{U}$ has a simple path of size n that is less than or equal to a given K. It now turns to prove that the above is true.

"$\Longrightarrow$"

Suppose that φ has a satisfying assignment $\vec{z} = \langle u_1 = a_1, u_2 = a_2, \ldots, u_m = a_m \rangle$, that is, $\varphi(\vec{z}) = 1$. Then, there exist partial assignments $\vec{x}_{j_i}$ in $\vec{z}$ such that, $\xi_i(\vec{x}_{j_i}) = 1, i \in [1, n]$. On the other hand, there also exist vertices C_{i,j_i} in $\mathfrak{U}$ associated to such $\vec{x}_{j_i}$. Therefore, it holds that:

$$\sum_{i=1}^{n} f(C_{i,j_i}) \leq K = \sum_{i=1}^{n-1} w(C_{i,j_i}, C_{i+1,j_{i+1}}) + 1. \tag{5.3.5}$$

"$\Longleftarrow$"

On the other hand, suppose that $\mathfrak{U}$ has a simple path $C_{1,j_1}, C_{2,j_2}, \ldots, C_{n,j_n}$. If it holds that:

$$\sum_{i=1}^{n} f(C_{i,j_i}) \leq K = \sum_{i=1}^{n-1} w(C_{i,j_i}, C_{i+1,j_{i+1}}) + 1; \tag{5.3.6}$$

then, this implies that the corresponding partial assignments $\vec{x}_{j_i}$ of C_{i,j_i} hold $\sum_{i=1}^{n} \xi_i(\vec{x}_{j_i}) =$ $= n \Rightarrow \xi_i(\vec{x}_{j_i}) = 1, \forall i \in [1, n] \Rightarrow \varphi(\vec{z}) = 1$, where $\vec{z}$ is the binary assignment vector from partial assignments $\vec{x}_{j_i}$. Hence finally PFBD belongs to the NP-complete problem class. □

Note that classifying PFBD as a NP-complete problem is not only important for our research. If a good solution is found for PFBD then this one would be also good for any other problem of the NP-complete class. Although NP-complete problems are computationally equivalent, they do no appear to be equivalent representing them through genetic algorithms [68]. That is, not all of these problems have a natural representation through genetic algorithms, and it could be quite difficult. As PFBD has a representation through genetic algorithms (given in the next chapter), then one strategy is to transform an NP-complete problem (which we generically call aNPCP) to PFBD, solving PFBD and transforming the solution back to aNPCP. Since the transformation of aNPCP to PFBD could also be difficult, we can use an intermediate problem as previous canonical problem SAT and the transformation process is as follows:

$$aNPCP \underset{solutionback}{\overset{transforms}{\rightleftarrows}} SAT \underset{solutionback}{\overset{transforms}{\rightleftarrows}} PFBD.$$

The problem transformation is from left to right, and transforming the solution back is in reverse form. Note also that the transformation between SAT and PFBD has previously been given, which facilities the task.

All we can say as a final conclusion of the PFBD NP-completeness is that, it is as hard as all those other problems that other scientists have been unable to solve. Hence, this is a good enough reason not to spend time and effort attempting to find a full solution. Instead, it would be better to make the effort to look for good approximate general solutions to specific instances of PFBD. In general aspects, the NP classification does help scientists decide where to invest their research efforts [76].

Chapter 6

An Evolutionary Model for Hierarchical Cluster Analysis

THIS chapter proposes a novel hierarchical clustering method using genetic algorithms (GAs) for the analysis of gene expression data. This method is based on the mathematical proof of several results, showing its effectiveness with regard to other clustering methods. GAs applied to cluster analysis have disclosed good results on biological data and many studies have been carried out in this sense, although most of them are focused on partitional clustering methods. Even though there are few studies that attempt to use GAs for building hierarchical clustering, they do not include constraints that allow us to reduce the complexity of the problem. Therefore, these studies become intractable problems for large data sets. On the other hand, the deterministic hierarchical clustering methods generally face the problem of convergence towards local optimums due to their greedy strategy. The method introduced here is an alternative to solve some of the problems existing methods face.

Firstly, since the convergence of most hierarchical clustering methods is local, due mainly to its greedy strategy for building a dendrogram [9, 10, 137], in this chapter we face the problem stated in the research hypothesis (Chapter 1), which is a combinatorial optimization problem ([155, 201]) that has been proven to be NP-complete from the previous chapter. It is common in computer science for problems of this class, to explore the use of heuristics in order to find approximate solutions to the problem

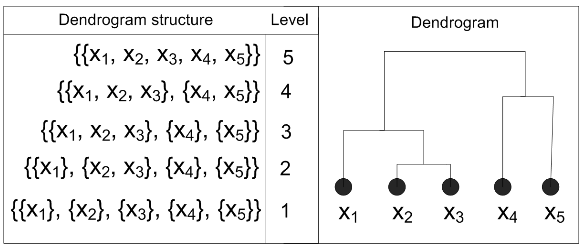
[70, 104, 243], or to converge faster towards some solutions. Exploring all possibilities of the dendrogram space is an intractable strategy, thus requiring methods that do not consider every solution in the search space, such as GAs [25, 105, 124, 185, 186].

Secondly, the practical interpretation of the problem previously stated in our hypothesis is to find a global optimum in the dendrogram space of DNA microarray data or in other words, to find dendrograms of high quality, which could disclose knowledge from *gene expression data*. However, it is difficult to build effective genetic operators on complex structures such as dendrograms, so this has been one of the problems faced in this chapter. Additionally, there is little literature on this subject, making it more difficult to solve, but we deal with these problems later.

6.1 The EMHC Genetic Algorithm

According to the previously said, the main contribution of this chapter is the definition of an evolutionary model aimed at building hierarchical clusterings. The goal of this approach is the search of clustering hierarchies of high quality on DNA microarray data. For reaching that aim, other contributions were given, such as: a specific method (called EMHC, Evolutionary Model for Hierarchical Cluster) from a clustering evolutionary model is obtained by prefixing the parameters of such a model. This method provides a novel fitness function to evaluate dendrograms based on the cluster definition. In this context, several strategies (constraints) are introduced to reduce the complexity of the search space. That is, reduction of the level number of a dendrogram based on non-valuable information, reduction of the fitness function runtime by introducing two fundamental lemmas, and the partition of the search space in neighborhoods to state differences between local and global optimum.

We also introduce two novel genetic operators (mutation and crossover) performing an agglomerative and divisive strategy to build the child dendrograms. In order to carry out in-depth search to improve the solutions given by our method, a novel evolutionary strategy of local search has been built. Another important result found from our experiments is that a genetic algorithm is not enough to deal with the problem of finding an optimum dendrogram in the search space. Hence, we have introduced several constraints and heuristics to make our problem tractable. All these contributions will allow our method finding better solutions than other ones.



Dendrogram structure	Level
$\{\{x_1, x_2, x_3, x_4, x_5\}\}$	5
$\{\{x_1, x_2, x_3\}, \{x_4, x_5\}\}$	4
$\{\{x_1, x_2, x_3\}, \{x_4\}, \{x_5\}\}$	3
$\{\{x_1\}, \{x_2, x_3\}, \{x_4\}, \{x_5\}\}$	2
$\{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}\}$	1

Figure 6.1: A dendrogram structure through sets (on the left side), representing each clustering of the hierarchy. The right side shows the associated dendrogram graph.

Finally, this chapter goes further than only defining an algorithm, such as in other researches. The idea consists of creating an evolutionary model where a set of parameters can be pre-fitted based on some criterium, so that we obtain a concrete clustering method able to adapt to the analyzed problem, but varying those parameters possibly, we will achieve a different method. Such an approach is possible through evolutionary computation.

6.1.1 Individuals and Encoding

The individuals (chromosomes) represent dendrograms on a given data set, encoded as an ordered set of clusterings, where each clustering is a ordered set of clusters. Every clustering of the dendrogram has an order number called level. Figure 6.1 shows a dendrogram on a data set with five elements, $\{x_1, x_2, x_3, x_4, x_5\}$. This hierarchical clustering is built in an agglomerative way and each level is represented by its corresponding number. That is, an individual is a dendrogram which consists of a collection of clusterings, where each clustering is a partition of the universe of objects to be clustered (a list of lists with subsets of the universe).

At first, each dendrogram of the initial population is built up from the first level to the higher level by joining two clusters randomly chosen in the current level to create the next one.

6.1.2 Length of an Individual

The dendrogram length is defined as its number of levels (clusterings). If the size of a data set is n, then the dendrogram length is $n - 2$, assuming that the first and the last level are not included. But in the best case, until the half of the dendrogram levels, there will be *unitary clusters* (one-element clusters) and that does not have practical meaning, hence, those levels can be removed. Following this reasoning, two or three-element clusters might not be also of interest and thus, a δ parameter can be introduced in order to remove the part of a dendrogram that does not give information from the user point of view. In other words, δ is the proportion of levels (a fraction, in mathematical terms) in a dendrogram that we want to remove from the first level, because it is assumed as noise (or non-valuable information). Therefore, the length of a dendrogram is defined as follows:

Definition 6.1. *Dendrogram length.*
Let $\mathfrak{P}_n$ be a data set of size n and let $\mathfrak{G}$ be a dendrogram on $\mathfrak{P}_n$, then the length of $\mathfrak{G}$ is the clustering number of it and is defined as:

$$|\mathfrak{G}| = n - 2 - \lfloor \delta \cdot n \rfloor, \tag{6.1.1}$$

where δ is the level proportion of $\mathfrak{G}$ that is removed, assuming $1/2 \leq \delta < 1$.

The reduction of the dendrogram length according to a user criterium of information validity is an efficient way of reducing both time and space in the execution of any hierarchical clustering method. In fact, there have been several works, such as those of [110, 115, 128, 211], on the estimation of the optimal number of clusters in a hierarchical clustering. Based on those, one can build dendrograms from the optimum number of clusters for the purpose of keeping a minimal number of levels. Nevertheless, this issue is not being considered in this research.

Based on Definition 6.1 we can give a proposition that relates the number of clusters that the level of a dendrogram has to the number of such a level (the level position within the dendrogram). The usefulness of this proposition will be seen later in Chapter 8.

Proposition 6.1. *The number of clusters in a level.*
Let *level* be the number of a level in dendrogram $\mathfrak{G}$ on $\mathfrak{P}_n$, and denote by k, the number of clusters in level *level* in $\mathfrak{G}$. Then, the following formula holds:

$$k = |\mathfrak{G}| - level + 2. \tag{6.1.2}$$

Proof: The proof of this proposition is easily reached by mathematical induction on the levels (*level*) of $\mathfrak{G}$. That is, starting with the last level by assigning $level = |\mathfrak{G}| \Longrightarrow k = 2$, repeating this process but decreasing *level* by one, the formula is proven. □

6.1.3 Fitness Function

In every genetic algorithm it is necessary to measure the goodness of candidate solutions. In this problem, the fitness of a dendrogram must be evaluated. In order to archive this, one of the given cluster definitions given in [8] has been considered, that is, *the objects inside a cluster are very similar, whereas the objects located in distinct clusters are very different.* Thereby, the fitness function is defined according to the concepts of *homogeneity* and *separation* introduced in [137]. We begin by defining cluster homogeneity and afterwards go on to define more complex structures until reaching the dendrogram structure.

Definition 6.2. *Cluster homogeneity.*
If $\mathfrak{D} = [d(i,j)]$ is the proximity matrix on the $\mathfrak{P}_n$ data set, being d a metric defined on this data set, $\mathfrak{C}$ a clustering of objects on $\mathfrak{P}_n$, C a cluster in $\mathfrak{C}$ and $m = |C|$, then the homogeneity of C is:

$$h(C) = \frac{2}{m \cdot (m-1)} \sum_{i \neq j}^{m\cdot(m-1)/2} d(i,j), (\forall i, j \in C). \tag{6.1.3}$$

Definition 6.3. *Clustering homogeneity.*
Let $\mathfrak{C}$ be a clustering of $\mathfrak{P}_n$, being $k = |\mathfrak{C}|$, then the homogeneity of $\mathfrak{C}$ ($\mathcal{H}(\mathfrak{C})$) is:

$$k \cdot \mathcal{H}(\mathfrak{C}) = \sum_{i=1}^{k} h(C_i) = \mathcal{H}^*(\mathfrak{C}), \tag{6.1.4}$$

$\mathcal{H}^*$ is called *absolute homogeneity.*

Definition 6.4. *Distance between two clusters.*
Let $\mathfrak{C}$ be a clustering of $\mathfrak{P}_n$, let C_1 and C_2 be two clusters of $\mathfrak{C}$, then the distance d_m between these clusters is defined as:

$$d_m(C_1, C_2) = \min\{d(i,j)/i \in C_1, j \in C_2\}. \tag{6.1.5}$$

Definition 6.5. *Clustering separation.*
Let $\mathfrak{C}$ be a clustering of $\mathfrak{P}_n$, let C_i and C_j be two different clusters of $\mathfrak{C}$, $k = |\mathfrak{C}|$, then the $\mathfrak{C}$ separation ($\mathcal{S}(\mathfrak{C})$) is:

$$\frac{k \cdot (k-1)}{2} \cdot \mathcal{S}(\mathfrak{C}) = \sum_{i \neq j}^{k \cdot (k-1)/2} d_m(C_i, C_j) = \mathcal{S}^*(\mathfrak{C}), \quad (\forall i, j \in [1, k]), \tag{6.1.6}$$

$\mathcal{S}^*$ is called *absolute separation.*

Since the definition of homogeneity and separation have been given, we can introduce the fitness functions of both a clustering and a dendrogram, that is:

Definition 6.6. *Clustering fitness function.*
Let $\mathfrak{C}$ and $\mathfrak{D}$ be a clustering of objects in $\mathfrak{P}_n$ and the proximity matrix of $\mathfrak{P}_n$ respectively, then the fitness function of $\mathfrak{C}$ is defined as:

$$f_c(\mathfrak{C}) = \frac{1}{k}\left[\frac{2 \cdot \mathcal{S}^*(\mathfrak{C})}{k-1} - \mathcal{H}^*(\mathfrak{C})\right] + \max \mathfrak{D}. \tag{6.1.7}$$

A fitness function should be positive by definition, thus, the term $\max \mathfrak{D}$ in the above sum avoids negative values of f_c.

Definition 6.7. *Dendrogram fitness function.*
If $\mathfrak{G}$ is a dendrogram on $\mathfrak{P}_n$ and $\mathfrak{C}_i$ a clustering of level i in $\mathfrak{G}$, then the fitness function of $\mathfrak{G}$ is:

$$f_d(\mathfrak{G}) = \frac{1}{|\mathfrak{G}|} \sum_{i=1}^{|\mathfrak{G}|} f_c(\mathfrak{C}_i), \tag{6.1.8}$$

where $|\mathfrak{G}|$ is the length of $\mathfrak{G}$ according to Definition 6.1.

Once defined the fitness function for the individuals, our goal is to maximize f_d by obtaining small values of $\mathcal{H}^*$ and large values of $\mathcal{S}^*$ for the clusterings of $\mathfrak{G}$.

Based on the above definition, we can now build an *agglomerative coefficient* (ac), which is used to estimate the cutoff level of a dendrogram whose corresponding clustering maximizes fitness function f_c.

Definition 6.8. *Agglomerative coefficient.*
Let $\mathfrak{G}$ and $\mathfrak{C}_i$ be a dendrogram on $\mathfrak{P}_n$ and a clustering of $\mathfrak{G}$, respectively. The agglomerative coefficient of $\mathfrak{G}$ is defined as:

$$ac(\mathfrak{G}) = \arg_{i\in[1,|\mathfrak{G}|]} \max f_c(\mathfrak{C}_i). \tag{6.1.9}$$

This *ac* coefficient is an alternative to the agglomerative and divisive ones defined in [10], facing problems of reaching large values when the data set grows. Furthermore, it could be used as an estimate of the optimum number of clusters in a data set.

Homogeneity of Unitary Clusters

Unitary clusters may not have practical meaning or they could be outliers or noise in the data set. Although the inclusion of the δ parameter in Definition 6.1 is understood to remove the unitary clusters of a dendrogram, these clusters can arise again as a result of the evolutionary process. Thus, since the homogeneity of unitary clusters cannot be assessed by Definition 6.2, it is necessary to define a homogeneity function for these clusters. Then, the homogeneity of a unitary cluster is performed by the inclusion of a new parameter α, which varies from 0% to 100% and acts as a penalization factor of unitary clusters. A homogeneity valor $p(\alpha)$ is assigned to α, where p is a penalization linear function whose values vary from $\min \mathfrak{D}$ to $\max \mathfrak{D}$ depending of the penalization factor ($p : [0, 100]\xrightarrow[linear]{}[\min \mathfrak{D}, \max \mathfrak{D}]$).

The penalization function is globally defined for all unitary clusters and checks the existence of unitary clusters in the whole dendrogram, restricting the existence of them. If the unitary clusters are discriminated using a large value of α in the execution of the algorithm, even though a unitary cluster still remains in a dendrogram of good fitness, then that cluster might be either an outlier or noise. Thereby, one could exclude the unitary clusters from cluster analysis and to repeat the process.

Finally, the homogeneity of a cluster C given in Definition 6.2, can be redefined as:

$$h'(C) = \begin{cases} p(\alpha), \text{if } |C| = 1; \\ h(C), \text{otherwise.} \end{cases} \tag{6.1.10}$$

6.1.4 Improving the Fitness Function Cost

Due to the computation complexity of the fitness function given in Definition 6.7, the need of decreasing its computation time has arisen. In the experiments on some data sets, the runtime of the fitness function turned out to be too long. From the theoretical point of view, the above is verified in the following proposition:

Proposition 6.2. *Algorithmic complexity of f_c.*
Let $\mathfrak{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ be a clustering of a dendrogram $\mathfrak{G}$ on $\mathfrak{P}_n$ and let f_c be the fitness function defined in (6.6), then the order of $f_c(\mathfrak{C})$ is $O(k^2m^2)$, where $m = \max\{|\mathcal{C}_i|, i \in [1,k]\}$.

Proof: We need only prove that $\mathcal{H}(\mathfrak{C})$ is $O(km^2)$ and $\mathcal{S}(\mathfrak{C})$ is $O(k^2m^2)$. That is, $\mathcal{H}^*(\mathfrak{C})$ depends of $h(C_i)$ (see Definitions 6.2 and 6.3) which computes m^2 steps, but h is applied to each cluster of $\mathfrak{C}$. Hence, $\mathcal{H}^*(\mathfrak{C})$ is $O(km^2)$. On the other hand, $\mathcal{S}^*(\mathfrak{C})$ depends of $d_m(C_i, C_j)$ (see Definitions 6.4 and 6.5) which computes m^2 steps, but it is run $\frac{k(k-2)}{2}$ times and so, $\mathcal{S}^*(\mathfrak{C})$ is $O(k^2m^2)$. Thus, $O(f_c) = \max\{O(km^2), O(k^2m^2)\} = O(k^2m^2)$.
□

The fitness function defined in (6.7), can be transformed in an equivalent but more efficient fitness function since only two clusters in a determined level will not be passing to the next level in a dendrogram. This is shown in the following lemmas:

Lemma 1. *Recurrent homogeneity.*
Let $\mathfrak{C}_i$ be a clustering of level i of a dendrogram $\mathfrak{G}$; C_j and C_l, the two clusters in $\mathfrak{C}_i$ such that its join forms a new clustering $\mathfrak{C}_{i+1}$ of next level $i+1$ in $\mathfrak{G}$. Then, the homogeneity of $\mathfrak{C}_{i+1}$ ($\mathcal{H}_1(\mathfrak{C}_{i+1})$) is computed in the following expression:

For $i = 1$, that is, for the first clustering, $\mathcal{H}_1(\mathfrak{C}_1) := \mathcal{H}(\mathfrak{C}_1)$.
For $i > 1$,

$$\mathcal{H}_1^*(\mathfrak{C}_{i+1}) = (k-1)\cdot\mathcal{H}_1(\mathfrak{C}_{i+1}) = k\cdot\mathcal{H}_1(\mathfrak{C}_i) - h'(C_j) - h'(C_l) + \frac{1}{\lambda_3}[\lambda_1\cdot h'(C_j) + \\ + \lambda_2\cdot h'(C_l) + l_1\cdot l_2\cdot d_p(C_j, C_l)], \tag{6.1.11}$$

where $k = |\mathfrak{C}_i|, l_1 = |C_j|, l_2 = |C_l|, \lambda_1 = \binom{l_1}{2}, \lambda_2 = \binom{l_2}{2}$, $\lambda_3 = \binom{l_1 + l_2}{2}$, d_p is the mean distance between objects of clusters C_j and C_l, $\mathcal{H}_1^*$ is the *recurrent absolute homogeneity.*

Proof: We need to prove that for all $t \in [1, |\mathfrak{G}|]$, the equality $\mathcal{H}_1(\mathfrak{C}_t) = \mathcal{H}(\mathfrak{C}_t)$ holds. We will do this by mathematical induction on levels t of the dendrogram $\mathfrak{G}$:

- for $t = 1$, the equality is true by definition of the lemma itself;
- let us suppose that the equality holds for $t = 2, 3, \ldots, i$, being $i < |\mathfrak{G}|$, that is, $\mathcal{H}_1(\mathfrak{C}_t) = \mathcal{H}(\mathfrak{C}_t)$;
- let us prove that the equality holds for $t = i+1$. From Definition 6.3 and equality (6.1.10), one has that:

$$\mathcal{H}(\mathfrak{C}_{i+1}) = \frac{1}{k-1} \sum_{z=1}^{k-1} h'(C_z), \tag{6.1.12}$$

taking the last term out of the sum, we obtain:

$$(k-1) \cdot \mathcal{H}(\mathfrak{C}_{i+1}) = \sum_{z=1}^{k-2} h'(C_z) + h'(C_{k-1}), \tag{6.1.13}$$

without losing generality, we can assume that $C_{k-1} = C_j \cup C_l$ and obtain:

$$\lambda_3 \cdot h'(C_j \cup C_l) = \lambda_1 \cdot h'(C_j) + \lambda_2 \cdot h'(C_l) + l_1 \cdot l_2 \cdot d_p(C_j, C_l), \tag{6.1.14}$$

on the other hand, we have that:

$$\sum_{z=1}^{k-2} h'(C_z) + h'(C_j) + h'(C_l) = k \cdot \mathcal{H}(\mathfrak{C}_i), \tag{6.1.15}$$

replacing the expressions (6.1.14) and (6.1.15) in (6.1.13), the equality is proven for $t = i+1$, hence it is true for all $t \in [1, |\mathfrak{G}|]$. □

Lemma 2. *Recurrent separation.*
Keeping the same conditions as in the previous lemma, one can obtain the recurrent separation of a clustering $\mathfrak{C}_{i+1}$ ($\mathcal{S}_1(\mathfrak{C}_{i+1})$):

For $i = 1, \mathcal{S}_1(\mathfrak{C}_1) := S(\mathfrak{C}_1)$.
For $i > 1$, we have that,

$$S_1^*(\mathfrak{C}_{i+1}) = (g-k+1)\cdot S_1(\mathfrak{C}_{i+1}) = g\cdot S_1(\mathfrak{C}_i) - d_m(C_j, C_l) - \textstyle\sum_{t\neq j\wedge t\neq l}^{k-2}[d_m(C_j,C_t) + \\ + d_m(C_l,C_t) - \min\{d_m(C_j,C_t), d_m(C_l,C_t)\}], \tag{6.1.16}$$

where $k = |\mathfrak{C}_i|, g = \binom{k}{2}$, being g the number of distances among the clusters of $\mathfrak{C}_{i+1}$, S_1^* is the *recurrent absolute separation.*

The proof of this lemma is similar to that of Lemma 1.

Definition 6.9. *Clustering recurrent fitness.*
The fitness function of a clustering $\mathfrak{C}_{i+1}$ of $\mathfrak{G}$, according to $\mathcal{H}_1^*$ and S_1^*, is defined as:

$$g_c(\mathfrak{C}_{i+1}) = \frac{S_1^*(\mathfrak{C}_{i+1})}{g-k+1} - \frac{\mathcal{H}_1^*(\mathfrak{C}_{i+1})}{k-1} + \max\mathfrak{D}, \tag{6.1.17}$$

known $\mathcal{H}(\mathfrak{C}_i)$ and $S(\mathfrak{C}_i)$. $S_1^*(\mathfrak{C}_{i+1})$ and $\mathcal{H}_1^*(\mathfrak{C}_{i+1})$ are computed from the formulas given in Lemmas 2 and 1 respectively.

Definition 6.10. *Dendrogram recurrent fitness.*
The fitness function of a dendrogram $\mathfrak{G}$, being $\mathfrak{C}_i$ a clustering of it is:

$$g_d(\mathfrak{G}) = \frac{1}{|\mathfrak{G}|-1}\sum_{i=1}^{|\mathfrak{G}|-1} g_c(\mathfrak{C}_i). \tag{6.1.18}$$

The goal of EMHC is maximizing g_d to achieve dendrograms of high quality by means of the genetic operators. Once defined the recurrences, one can verify that the cost of the fitness function defined in (6.3.8) is less than the cost of the fitness function defined in (6.1.7). This is shown in the following proposition:

Proposition 6.3. *Algorithmic complexity of g_c.*
Let $\mathfrak{C}_i, \mathfrak{C}_{i+1}$ be two clusterings (levels i and $i+1$) of a dendrogram $\mathfrak{G}$ on $\mathfrak{P}_n$, $k = |\mathfrak{C}_i|$ and $m = \max\{|C_j|, j \in [1,k]\}$, then the temporal complexity of $g_c(\mathfrak{C}_{i+1})$ is $O(km^2)$.

Proof: We need only prove that $\mathcal{H}_1^*(\mathfrak{C}_{i+1})$ is $O(m^2)$ and $S_1^*(\mathfrak{C}_{i+1})$ is $O(km^2)$. That is, the function with more complexity in the expression of $\mathcal{H}_1^*$ in Lemma 1 is d_p, which computes in $O(m^2)$. On the other hand, the function with more complexity in the expression of $S_1^*(\mathfrak{C}_{i+1})$ in Lemma 2 is $d_m(C_j, C_t)$ (or $d_m(C_l, C_t)$), which computes in $O(m^2)$, but it is run $k-2$ times, implying that $S_1^*(\mathfrak{C}_{i+1})$ is $O(km^2)$. Finally, $O(g_c) = \max\{O(m^2), O(km^2)\} = O(km^2)$. $\square$

Therefore, the computations with fitness function g_d are more efficient than f_d. Moreover, we experimentally observe that the runtime of f_d was decreased by a factor 5 of 1/5 using g_d. On the other hand, although g_d is more efficient, f_d is still important, because it is an independent function of the inter-cluster distance type, which is used for assessing $\mathcal{H}^*$ and $\mathcal{S}^*$. This does not stand for g_d that uses $\mathcal{H}_1^*$ and $\mathcal{S}_1^*$.

6.1.5 Mutation Operator

Our first mutation operator (FMO) is a unitary alteration operator which only affects a single individual. It only transforms a part of a dendrogram, exploring different branches and returning a new dendrogram in order to replace the previous one. Overall, the FMO carries out an in-depth search. The mutation of a dendrogram is performed according to the following steps:

1. Consider two parameters τ and ϵ for each dendrogram $\mathfrak{G}$ where:
   - τ is the choosing percentage of cluster pairs in level i to build the following level $i+1$;
   - ϵ is a small value that represents the similarity between two clusters according to the homogeneity measure.
2. A random number $i \in [1, |\mathfrak{G}| - 1]$ is generated to choose the level where the mutation of $\mathfrak{G}$ is carried out.
3. For the clustering of level i of the previous step, one of the following conditions is chosen:
   - the most homogeneous join of cluster pairs of $\tau\%$ of random cluster pairs is chosen;

- choose a new cluster pair whose homogeneity difference with the cluster pair chosen in the above condition be less than or equal to ϵ.

4. The cluster pair chosen in the previous step is joined in order to form a new cluster so that the clustering of the next level $i + 1$ is built.

5. Steps 3 and 4 are repeated on level $i + 1$, until it reaches the last level.

Note that only a part of dendrogram $\mathfrak{G}$ is modified, the another part is kept unchangeable. On the other hand, the search for the best cluster pair in a level of $\mathfrak{G}$ is controlled by τ and the search for the cluster pairs similar to the best cluster pair is controlled by ϵ, which guarantees good clusters for the next levels.

The possibility, step 3, of choosing two new clusters to be merged in step 4 of the mutation operator algorithm is similar as exchanging two branches of a tree in *genetic programming* [28, 29, 42, 150, 162], since a dendrogram can be considered as a tree, Figure 6.1.

Figure 6.2-(A) summarizes, through a hypothetical example, the steps previously given by the FMO. As shown, the dendrograms have been built on data set $\{a, b, c, d, e\}$. Dendrogram #1 shows (in red color) the clusters selected to be joined in order to build the next upper level. In this case, level 3 is selected to carry out the mutation process and create Dendrogram #2. Then, Dendrogram #2 is created from level 3 in Dendrogram #1 by making a different selection of clusters on this level (clusters in red color, Dendrogram #2). The process is repeated on the new level (in this case, level 4) until reaching the last level. As shown in Dendrogram #2, it has been modified only levels from 3 to 5 in Dendrogram #1. Note that Figure 6.2-(B) shows the same process as in Figure 6.2-(A) but in form of dendrogram graphs.

A very important concept associated with the FMO is the *neighborhood* concept, [17, 90, 118, 135]. In the dendrogram search space from a data set, there potentially is a great number of global optima, but there are many more locally optimum solutions. Thereby, considering the neighborhood of a solution allows us a systematic approach for the distinction between local and global optima.

Definition 6.11. *Neighborhood of a dendrogram.*
Let $\mathfrak{G}$ and $\mathfrak{G}'$ be two dendrograms on $\mathfrak{P}_n$, let $\mathfrak{C}_1$ be a clustering of the first level of the $\mathfrak{G}$ and let $\mathfrak{C}'_1$ be a clustering of the first level of $\mathfrak{G}'$, then the neighborhood of $\mathfrak{G}$ is

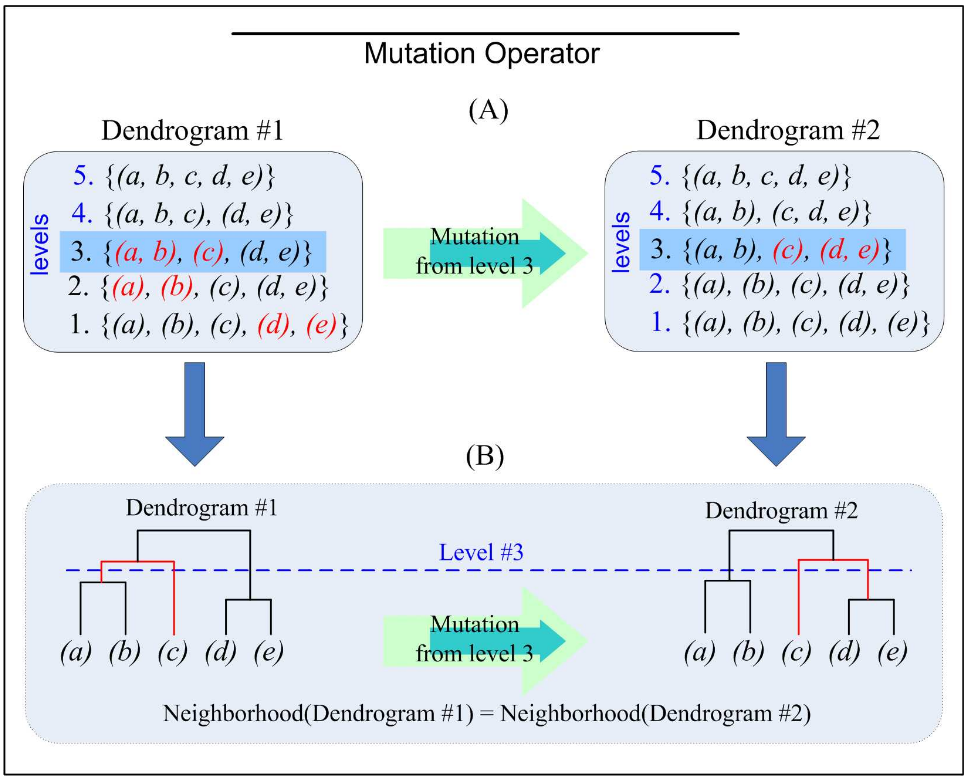


Figure 6.2: A hypothetical example summarizing the running of the mutation operator defined for the EMHC method. (A) shows the mutation process from the internal structure of the dendrograms whereas (B) shows the same, but from the graph of the dendrograms.

defined as:

$$V(\mathfrak{G}) = \{\mathfrak{G}' \,|\, \mathfrak{C}_1 = \mathfrak{C}'_1\}, \tag{6.1.19}$$

that is, the set of all dendrograms $\mathfrak{G}'$ whose clusters in the first level match the ones in the first level of $\mathfrak{G}$.

Since the FMO does not modify the first level of a dendrogram, neighborhoods can be built from the individuals whose first level clusterings share the same clusters (any other transformation that modifies the first level yields a different neighborhood). Based on the above definition one can introduce the size of a neighborhood, namely the number of dendrograms belonging to a neighborhood.

Proposition 6.4. *Size of the neighborhood of a dendrogram.*
Let $\mathfrak{G}$ be a dendrogram on $\mathfrak{P}_n$ whose first level has $k = |\mathfrak{G}| + 1$ clusters, then the size of the neighborhood of $\mathfrak{G}$ is:

$$|V(\mathfrak{G})| = \frac{k(k-1)!^2}{2^{k-1}}. \tag{6.1.20}$$

Proof: $\binom{k}{2}$ different cluster pairs can only be chosen from the first level of $\mathfrak{G}$ so that level 2 can be built. Repeating this process on level 2 and following this idea until reaching the last level, one obtains the expression below:

$$|V(\mathfrak{G})| = \prod_{i=0}^{k-2} \binom{k-i}{2} = \frac{1}{2^{k-1}} \prod_{i=0}^{k-2} (k-i)(k-i-1), \tag{6.1.21}$$

and the expected result is obtained by expanding the right side of the equality. □

The precedent proposition is an important result according to which heuristics can be added to the search process and moreover, the values of the mutation and crossover probability, as well as τ and ϵ parameters can be assigned according to the size of the neighborhood. Alternatively, the dendrograms generated by applying the FMO form a neighborhood where each dendrogram depends on the first level of the transformed dendrogram. Therefore, the result returned by the FMO can lead us to the best dendrogram in a neighborhood; in other words, it converges to a local optimum on the search space. This way, arises the need of using a recombination operator that performs between neighborhoods rather than within them as shown in the next subsection.

To conclude, an optional second mutation operator (SMO) can be defined, so that it could be used in a local search strategy or in combination with the FMO. The SMO on a dendrogram $\mathfrak{G}$ is built as follows:

1. Choosing a random level $i \in [1, |\mathfrak{G}| - 1]$.
2. Choosing two random cluster positions (s, t) in the clustering of level i.
3. Choosing an element s_α in cluster s and an element t_β in cluster t randomly.

4. Exchanging elements s_α and t_β of clusters s and t in level i, respectively.

5. Spreading the modification of the above step to clusters of the remaining levels different of level i for which s_α and t_β belong.

6.1.6 Crossover Operator

The crossover operator (CO) recombines valuable information of two or more individuals in order to build new individuals, which inhere the genetic code of their ancestors. The CO carries out a wide search in the dendrogram space. The crossover is based on the idea in [109] and [108], and performs on two father dendrograms to obtain a child dendrogram, that is:

1. Given two dendrograms $\mathfrak{G}_1$ and $\mathfrak{G}_2$ (parents), a random number i in $[1, |\mathfrak{G}_1|]$ is generated to choose the level where one can carry out the crossover between both dendrograms.

2. Through a strategy of *greedy algorithm* (looking up [19, 97, 247]), the best $\lfloor k/2 \rfloor$ clusters (the most homogenous clusters of each dendrogram) of level i in each dendrogram of the above step are chosen, being k the number of clusters in level i. A new clustering is formed by repairing the chosen clusters to avoid repeated object data [46, 109].

3. As soon as the new clustering for level i is built, one can build up the new dendrogram:

   - levels higher than level i are built using the FMO;
   - levels lower than level i are built in a divisive way, that is, for each level less than i, the less homogenous cluster is chosen to be split into two. This process is repeated until reaching the first level of the dendrogram.

4. The parent of less fitness value is replaced by the child dendrogram of step 3 by the replacement stage of EMHC.

Figure 6.3-(A) summarizes, through a hypothetical example, the steps of the CO. As shown, the used data set is $\{a, b, c, d, e, f, g\}$. From two hypothetical dendrograms (the parents) on this data set, a level is randomly selected (level 4 for this example) and the two clusterings (i and ii) of this level in both dendrograms are taken out as shown in this figure (Step #1). The most homogenous clusters (represented in green color, clustering i and ii) are selected for each clustering in order to create an intermediate clustering, which has repeated and missing object data (Step #2). Thus, this clustering is repaired to obtain a new clustering being the seed of the child dendrogram. With the seed clustering, the child dendrogram is built (Step #3) by applying the MO to form the upper levels to level 4 (starting by joining clusters (f) and (b) from the seed level). The lower levels to level 4 of the child dendrogram are formed by applying a divisive strategy (starting by splitting cluster (d, e, g) into clusters (d, e) and (g) from the seed level). Note that in Figure 6.3-(B) has shown both parent-dendrograms and the result of the crossover (child-dendrogram) in form of dendrogram graphs.

This CO has an important characteristic, namely, the new individual built from the parents does not belong to the neighborhoods of its parents, meaning that this operator explores regions in the search space where the FMO is not able to explore. Hence, the CO is also responsible for the diversity of the individuals in each generation of the GA. Based on this idea the following proposition is introduced:

Proposition 6.5. *Minimal number of individuals.*
If $\mathfrak{U}$ is the universe set of all the dendrograms (the search space) on $\mathfrak{P}_n$ (a data set), then $\mathfrak{U}$ can be generated with a population of at least two dendrograms according to the defined genetic operators.

Proof: It is directly proven from the definitions of the mutation and crossover operators. That is, with one individual, it can only use the mutation operator (FMO), which performs only within the neighborhood of the initial individual. Thus, it can not cover the whole $\mathfrak{U}$. Hence, we need two individuals (as a minimum) to use the crossover operator (CO), which performs inter-neighborhoods, generating individuals in new neighborhoods to cover the whole $\mathfrak{U}$. $\square$

The previous proposition assures that two individuals are needed as a minimum to cover the whole $\mathfrak{U}$. This is not true in the case of a unique individual, the FMO or other problems that deal with GAs. Theoretically, the above states that if the $\mathcal{GA}$ genetic algorithm with the genetic operators defined previously is considered as a black box including both t generations and two individuals $\mathfrak{G}_1, \mathfrak{G}_2$; then:

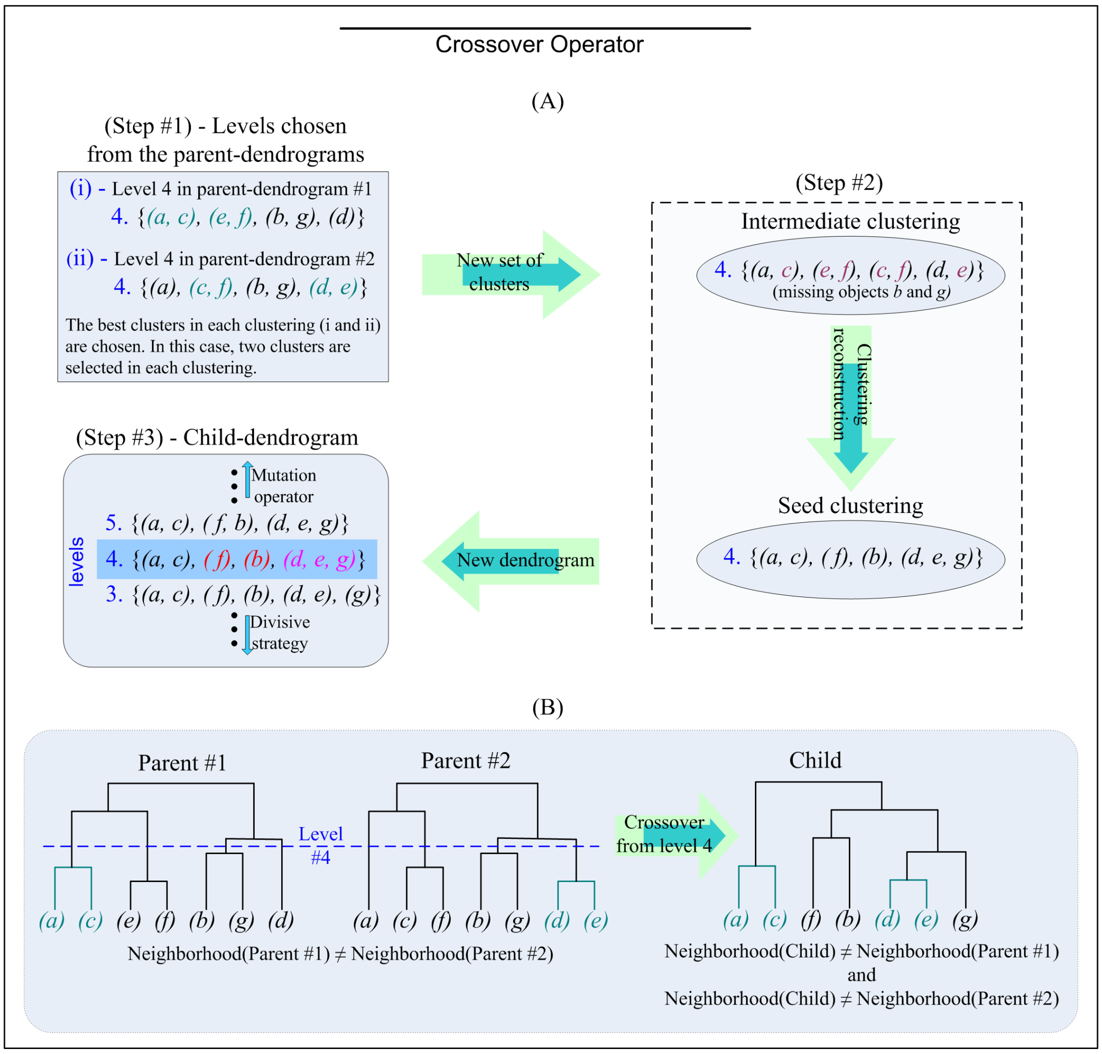

Crossover Operator
(A)
(Step #1) - Levels chosen from the parent-dendrograms
(i) - Level 4 in parent-dendrogram #1
4. {(a, c), (e, f), (b, g), (d)}
(ii) - Level 4 in parent-dendrogram #2
4. {(a), (c, f), (b, g), (d, e)}
The best clusters in each clustering (i and ii) are chosen. In this case, two clusters are selected in each clustering.
New set of clusters
(Step #2)
Intermediate clustering
4. {(a, c), (e, f), (c, f), (d, e)}
(missing objects b and g)
Clustering reconstruction
Seed clustering
4. {(a, c), (f), (b), (d, e, g)}
New dendrogram
(Step #3) - Child-dendrogram
Mutation operator
5. {(a, c), (f, b), (d, e, g)}
4. {(a, c), (f), (b), (d, e, g)}
3. {(a, c), (f), (b), (d, e), (g)}
Divisive strategy
levels
(B)
Parent #1
Parent #2
Child
Level #4
Crossover from level 4
(a) (c) (e) (f) (b) (g) (d)
(a) (c) (f) (b) (g) (d) (e)
(a) (c) (f) (b) (d) (e) (g)
Neighborhood(Parent #1) ≠ Neighborhood(Parent #2)
Neighborhood(Child) ≠ Neighborhood(Parent #1)
and
Neighborhood(Child) ≠ Neighborhood(Parent #2)

Figure 6.3: A hypothetical example summarizing the running of the crossover operator defined for the EMHC method. (A) shows the crossover process from the internal structure of the dendrograms whereas (B) shows the same, but from the graph of the dendrograms.

$$\lim_{t\to\infty} \mathcal{GA}(t, \{\mathfrak{G}_1, \mathfrak{G}_2\}) = \mathfrak{G}, \tag{6.1.22}$$

where $\mathfrak{G}$ is a dendrogram which is a global optimum in $\mathfrak{U}$.

In addition, it is possible to define an equivalence relation (a reflexive, symmetrical and transitive relation) on $\mathfrak{U}$ from Definition 6.11, namely, considering the binary relation *"to be in the neighborhood of"*. Therefore, $\mathfrak{U}$ is partitioned by this relation into disjoint neighborhoods. In other words, the neighborhoods do not intersect each other and every individual in a population determines one and only one region of $\mathfrak{U}$. This explanation supports Proposition 6.5 and expression (6.3.2).

Going back to the CO, it can optionally be improved by introducing an strategy performing on the clustering returned in step 2 of the CO. That is, the information that could be lost in the clustering reconstruction process in step 2 of the CO can be retrieved using an evolutionary search strategy. First, we define two operators to modify a clustering:

- ALT1 operator: given a clustering, it exchanges two elements between two clusters. Both clusters and elements are chosen randomly.
- ALT2 operator: given a clustering, it moves one element from a cluster to another cluster. Both clusters and the element are chosen randomly.

The evolutionary strategy to improve the clustering returned by step 2 of the CO can be defined as follows:

6.1.7 EvolCluster Algorithm

Input: $\mathfrak{C}$, a clustering. OP, a operator performing on $\mathfrak{C}$, which can be ALT1 or ALT2. MaxGeneration, the number of iterations. EvalFitness evaluates the fitness of a clustering $\mathfrak{C}$.

Output: $\mathfrak{C}$, improved.

```
1. % Computing the fitness of $\mathfrak{C}$.
2. $f$ := EvalFitness($\mathfrak{C}$); $t$ := 0;
```

```
3. while $t <=$ MaxGeneration do
4.    $t := t + 1$;
5.    % Applying an alteration.
6.    $\mathfrak{C}' :=$ OP($\mathfrak{C}$);
7.    % Evaluating the new clustering.
8.    $f' :=$ EvalFitness($\mathfrak{C}'$);
9.    % Updating of the improved clustering.
10.   if $f' > f$ then
11.      $f := f'; \mathfrak{C} := \mathfrak{C}'$
12.   endif
13. endwhile
14. end.
```

Finally, note that the SMO can be classified as a genetic operator intermediate between the FMO and the CO, because it carries out an alteration whose output individual fall within a distinct neighborhood of the altered individual. In other words, the output individual is in a neighborhood close to the one of the altered individual, much closer than an output individual generated by the CO.

6.2 Execution Strategy of EMHC

The execution steps of the EMHC method are the same steps as the general scheme of a genetic algorithm (GA), which is listed in [25, 105, 118], but in our case, an elitism for each generation, the roulette wheel selection method and immediate replacement of individuals were included in the evolutionary process.

In order to obtain better performance in the behavior of EMHC, the power of the GA and the speed of a local optimizer were merged. This goal was fulfilled based on one of the strategies proposed in [118], that is:

- Running a GA until it slows down, then letting a local optimizer take over the last generation (and/or best individual) of the GA. Hopefully, the GA is very close to the global optimal.

The GA execution in its early stage guides the individuals of each generation toward the neighborhood of some global optimum. In the second stage, the local optimization strategy is liable for searching a global optimum. There are many local optimization strategies ([17, 118]) but here, a hybrid evolutionary search process has been introduced according to the mutation operator. The following search strategy is proposed:

6.2.1 LocalSearch Algorithm

Evolutionary strategy to improve the output of EMHC.

Input: POP, the population composed of the individuals of the last generation of EMHC. MaxGeneration, the number of iterations. MO, a mutation operator. EvalFitness, fitness function of a dendrogram.

Output: POP, as a result of improvement of the input.

```
1. $i \in [1, |POP|]$;
2. % Computing the fitness of each individual.
3. for all $p.i$ in POP do
4.     $f.i$ := EvalFitness($p.i$);
5. endfor
6. $t := 0$;
7. while $t <=$ MaxGeneration do
8.     $t := t + 1$;
9.     for all $p.i$ in POP do
10.        % Applying mutation.
11.        newp := MO($p.i$);
12.        % Evaluating the new individual.
13.        newf := EvalFitness(newp);
```

```
14.         % Updating of the improved individual.
15.         **if** newf > $f.i$ **then**
16.           $p.i$ := newp; $f.i$ := newf
17.         **endif**
18.     **endfor**
19. **endwhile**
20. **end.**
```

Note that this strategy can be used not only to improve a population but also to improve an individual. Moreover, we can use mutation operators and fitness functions different of the defined.

6.3 Adding a New Objective to the Fitness Function

The fitness function introduced by Definition 6.9 depends of two objectives, homogeneity and separation, but this function does not check the distribution of data in the clusters of the given clustering. This aspect could be important in cluster analysis for determined problems. In the particular case of our approach, which is based on an agglomerative and a divisive strategy to build dendrograms, allowing it to find good small and large clusters at a time. On one hand, the above is a good result because EMHC finds solutions that can not be reached by other methods and in addition, data in small clusters might be outliers or noise. On the other hand, the data number distributed in each cluster of a clustering may be very different. In order to reduce large differences in the size of the clusters of a clustering, an optional objective can be defined as follows:

Definition 6.12. *Clustering deviation with respect to a $\hat{t}$ parameter.*
Let $\mathfrak{C} = \{C_1, C_2, \cdots, C_k\}$ be a clustering of $\mathfrak{P}_n$, $\mathfrak{M} = \{m_1 = |C_1|, m_2 = |C_2|, \cdots, m_k = |C_k|\}$. The deviation of the size of the clusters in $\mathfrak{C}$ with respect to a $\hat{t}$ parameter (called *clustering deviation with respect to $\hat{t}$*) is:

$$\sigma^2(\mathfrak{C}) = \frac{1}{k}\sum_{i=1}^{k}\left(m_i - \hat{t}\right)^2, \hat{t} = \lfloor n/k \rfloor, i \in [1, k]. \tag{6.3.1}$$

This clustering deviation is used to penalize clusterings whose clusters are very different in size, that is, clusterings with small and large clusters. Thus, it can be included in fitness function (6.3.8) by transforming the minimum problem of (6.3.1) to a maximum one. To do that, we need a higher bound of σ and so the following theorem exposes that $\sigma(\mathfrak{C})$ is bounded:

Theorem 6.1. *σ is a bounded function.*
If $\mathfrak{C} = \{C_1, C_2, \cdots, C_k\}$ is a clustering of $\mathfrak{P}_n$, σ the clustering deviation defined on $\mathfrak{C}$ in (6.3.1), then $\sigma(\mathfrak{C})$ is bounded and satisfies that:

$$0 \leq \sigma(\mathfrak{C}) < \beta(k) = \sqrt{\frac{(k-1)\hat{t}^2 + (n-\hat{t})^2}{k}}. \tag{6.3.2}$$

Proof:

- It is easy to see that $0 \leq \sigma(\mathfrak{C})$ holds;
- We now prove the right side of the inequation, namely $\sigma(\mathfrak{C}) < \beta(k)$:

$$\sum_{i=1}^{k} \left(m_i - \hat{t}\right)^2 < (k-1)\hat{t}^2 + (n-\hat{t})^2, \tag{6.3.3}$$

$$\sum_{i=1}^{k} \left(m_i - \hat{t}\right)^2 < (k-1)\hat{t}^2 + \left(\sum_{i=1}^{k} m_i - \hat{t}\right)^2, \tag{6.3.4}$$

$$\sum_{i=1}^{k} m_i^2 - \cancel{2\hat{t}\sum_{i=1}^{k} m_i} + k\hat{t}^2 < (k-1)\hat{t}^2 + \left(\sum_{i=1}^{k} m_i\right)^2 - \cancel{2\hat{t}\sum_{i=1}^{k} m_i} + \hat{t}^2 \tag{6.3.5}$$

$$\text{and finally } \sum_{i=1}^{k} m_i^2 < \left(\sum_{i=1}^{k} m_i\right)^2 \text{ holds.}\square \tag{6.3.6}$$

At this point, difference $\beta(k) - \sigma(\mathfrak{C})$ can be added as a new objective of the fitness function in (6.3.8). On the other hand, based on the proof of theorem 6.1, we can prove that the order of $\sigma(\mathfrak{C})$ is $O(m^2)$, where $m = \max \mathfrak{M}$. However, this order can be improved for the clusterings of a dendrogram focused on the reasoning given in the proof of Lemma 1:

Lemma 3. *Clustering recurrent deviation.*
Let $\mathfrak{C}_i$ be the clustering of level i and let $\mathfrak{C}_{i+1}$ be the clustering of level $i+1$, both in a dendrogram $\mathfrak{G}$ of $\mathfrak{P}_n$; C_j and C_l, two clusters of level i such that its join forms the new clustering $\mathfrak{C}_{i+1}$ of level $i+1$, then the recurrent deviation of $\mathfrak{C}_{i+1}$ ($\sigma_1(\mathfrak{C}_{i+1})$) is computed as:

if $i=1$, $\sigma_1(\mathfrak{C}_1) := \sigma(\mathfrak{C}_1)$ and for $i>1$,

$$\begin{aligned}&\sigma_*^2(\mathfrak{C}_{i+1}) = (k-1)\cdot\sigma_1^2(\mathfrak{C}_{i+1}) = k\cdot\sigma_1^2(\mathfrak{C}_i) + (m_j+m_l)^2 + 2\cdot n\cdot(\hat{t_i}-\hat{t}_{i+1}) + \\ &+ \hat{t}_{i+1}^2\cdot(k-1) - m_j^2 - m_l^2 - k\cdot\hat{t}_i^2,\end{aligned} \tag{6.3.7}$$

where $k = |\mathfrak{C}_i|, m_j = |C_j|, m_l = |C_l|, \hat{t_i} = \lfloor n/k\rfloor, \hat{t}_{i+1} = \lfloor n/(k-1)\rfloor$, σ_* is called *recurrent absolute deviation* of $\mathfrak{C}_{i+1}$.

Note $\sigma_1(\mathfrak{C}_{i+1}) = \sigma(\mathfrak{C}_{i+1})$, but σ_1 is more efficient in runtime, in fact, it is not difficult to prove that σ_1 is $O(m)$, where $m = \max\mathfrak{M}_{i+1}$. To conclude, the new fitness function from Definition 6.9 is defined as:

Definition 6.13. *Clustering fitness with recurrent deviation.*
The fitness function of a clustering $\mathfrak{C}_{i+1}$ of $\mathfrak{G}$, according to $\mathcal{H}_1^*$, S_1^* and σ_* is defined as:

$$f_{rd}(\mathfrak{C}_{i+1}) = \frac{S_1^*(\mathfrak{C}_{i+1})}{g-k+1} - \frac{\mathcal{H}_1^*(\mathfrak{C}_{i+1})}{k-1} + \left[\beta(k-1) - \frac{\sigma_*(\mathfrak{C}_{i+1})}{\sqrt{k-1}}\right] + \max\mathfrak{D}, \tag{6.3.8}$$

known $\mathcal{H}(\mathfrak{C}_i)$, $S(\mathfrak{C}_i)$ and $\sigma_1(\mathfrak{C}_i)$. $S_1^*(\mathfrak{C}_{i+1})$, $\mathcal{H}_1^*(\mathfrak{C}_{i+1})$ and $\sigma_*(\mathfrak{C}_{i+1})$ are computed from the formulas given in Lemmas 2, 1 and 3 respectively.

Definition 6.14. *Dendrogram fitness with recurrent deviation.*
The fitness with recurrent deviation of a dendrogram $\mathfrak{G}$, being $\mathfrak{C}_i$ a clustering in it is:

$$g_{rd}(\mathfrak{G}) = \frac{1}{|\mathfrak{G}|-1}\sum_{i=1}^{|\mathfrak{G}|-1} f_{rd}(\mathfrak{C}_i). \tag{6.3.9}$$

6.4 Implementing EMHC

EMHC was implemented and tested under the *R language* (R Development Core Team [16], using the packages in [56, 176]), which is a free distribution project providing an environment for statistical computation that couples graphic tools. R offers a wide variety of statistical techniques, namely, linear and nonlinear modeling, statistical tests, time series analysis, classification and clustering among others. Hence, R as a programming language be highly desirable for data and cluster analysis.

R project is the result of contributions of many collaborators from around the world, for which, trough of including computational packages make R a powerful system of data analysis. Of particular interest are the contributions of the *bioconductor project*, which is a compendium of packages on R aimed at the tool development for Bioinformatics research. Bioconductor is specifically designed for the analysis and understanding of genomic data as DNA microarray data.

Finally, EMHC is distributed in the R package format (the package is called *clustergas*) to be installed in the system. In addition to the EMHC method, this package includes a set of utilitarian functions divided into genetic operators, fitness functions, cluster validity measures and data treatment functions, among others. The package and its reference manual are freely available (under *R-Project* licence) at `http://cran.r-project.org/web/packages/clustergas`.

Chapter 7

Theory of the Visual Framework

THIS chapter presents the theory of the visual analytics framework to be used in cluster analysis from gene expression data. In addition, this approach provides a novel method of finding cluster boundaries based on the theory of metric spaces. The tool built as a result of this approach links a set of visualizations able to interact with parallel coordinates, cluster boundary points on a 3D scatter plot (using dimensionality reduction, Appendix B) and DNA microarray visualizations. Thus, it is also a visual alternative with respect to cluster validity measures currently used. Besides that, the method of computing cluster boundary is also used to estimate the shape that a cluster has on a 3D-space, and represent reference partitions (on a 3D-space) coming from the problem domain. Therefore, visual data exploration has been introduced for aggregating, summarizing and visualizing information generated during interactive cluster analysis from DNA microarray data [8–11]. Based on the above, we have developed a prototype tool called *3D-VisualCluster* (3D-VC), which has been presented at the end of this chapter.

7.1 Boundary Points: Background

Boundary points are data points that are located at the margin of densely distributed data, and are very useful in data mining applications since they represent a subset of the population that possibly belongs to two or more classes [250]. Awareness of these points is also useful in classification tasks, since they can potentially be misclassified [9].

Cluster boundary reconstruction is of great importance in DNA microarray data analysis, since:

- The boundary genes of a cluster may be representative of the class that this cluster determines, and so interior genes can be discriminated by the boundary genes [137];
- The above approach may disclose additional knowledge about the functions of many genes and raises hypotheses regarding mechanisms in the transcriptional regulatory network [66];
- Surface reconstruction bounded by boundary gene-points produces shapes and structures that may be meaningful in the context of gene expression data cluster analysis (*pattern recognition*), that otherwise would not have been possible [21,82].

According to [156,250], a boundary point p is an object that meets the following conditions:

a) It is within a dense region $\mathbb{R}$;

b) There exists a region $\mathbb{R}'$ near p such that $Density(\mathbb{R}') \gg Density(\mathbb{R})$ or $Density(\mathbb{R}') \ll Density(\mathbb{R})$, where the density of a region ($Density(\mathbb{R})$) measures the relative number of points it contains with respect to its size.

Based on these conditions, in [250] was developed a method that uses the technique of reverse k nearest neighbor (RkNN) [154]. Using RkNN on a data set requires the execution of a query for each point in the data set. Thus, this is an expensive task with complexity $O(n^3)$, where n is the size of the data set [236]. On one hand, this is a complex method that is applied to a whole data set rather than of a cluster. On the other hand, although this method performs well, it is intended to separate dense regions from less dense ones, and therefore implicitly performs a clustering task.

Our method of boundary points is oriented to perform on clusters, and thus the goals differs from those of the previous strategy. That is, both strategies are not comparable since our method assumes that the data has previously been grouped into clusters. Furthermore, our method is based on the boundary definition in terms of theoretical notions from *metric spaces* (see Appendix C). This way, boundary points focus on the set of points at the closure of a cluster that do not belong to the interior of the cluster, as will be shown later. Finally, our method runs in $O(k^2)$ time, where k is the size of the cluster, which makes it less complex and more suitable for an interactive framework.

7.2 Metric-based Cluster Boundaries and Surfaces

To support the visual framework, we first have developed an approach of analysis from DNA microarray data focused on metric spaces, which allowed us introducing an algorithm of finding cluster boundary gene-points. Based on boundary gene-points, it is possible to build the approximated surface of clusters as well as represent a reference partition in the space.

These new visualizations are combined with other DNA microarray visualizations, such as: heat maps, dendrograms, parallel coordinates, that our framework provides through linked views. This way, the visual analytics process is reached. On the other hand, as our prototype is linked to the *R language* [16], the results reached by clustering methods implemented on R can be displayed by the 3D-VC tool. Since R is a programming language widely used in Bioinformatics, almost all clustering methods are developed on R through software packages.

To carry out all the above, we start by considering the cluster problem as *finding connected regions in a multi-dimensional space containing a relatively high density of points, separated from other such regions by a region containing a low density of points* [8], and assuming that the objects to be clustered are represented as points in the space $\mathbb{R}^d$.

Since $\mathbb{R}^d$ with a defined metric (usually the Euclidean distance) is a *metric space* [159, 221], it can also be assumed the gene expression matrix of a DNA microarray as a subspace of $\mathbb{R}^d$. Starting from these concepts, some important definitions and conditions are given before we present an algorithm which computes the boundary points of a given cluster. This algorithm also improves the runtime of the cluster surface reconstruction algorithm implemented by the proposed visual framework, 3D-VC.

7.2.1 Boundary Points of a Cluster

We assume that the gene expression matrix is a bounded metric space as stated in proposition D.1 of the Theoretical Results (Appendix D). This allows the domain in which we will work to be defined. Next, the cluster problem is accordingly defined on the bounded metric space. Finally, a proposition about whether a cluster is opened or closed is presented. This proposition is useful for defining the problem of cluster boundary points.

In what follows, the set $\mathfrak{G}_n^d$ denotes a bounded subspace of $\mathbb{R}^d$ with an induced metric ρ, where d is the dimension and n is the number of genes. That is, $\mathfrak{G}_n^d$ with the metric ρ is a bounded discrete metric space, called the *gene metric space*. The methods introduced in this section are based on definitions and proofs given in the sections entitled Metric Spaces and Theoretical Results (Appendices C and D respectively).

We can now state the cluster problem on $\mathfrak{G}_n^d$, which is defined according to Chapter 3:

Definition 7.1. *Cluster problem on $\mathfrak{G}_n^d$.*
Let $\mathfrak{G}_n^d$ be a metric space. Then a partition $\mathfrak{C}$ of $\mathfrak{G}_n^d$ is a collection of subsets $\{C_1, C_2, \ldots, C_m\}$ of $\mathfrak{G}_n^d$ satisfying:

$$C_i \cap C_j = \emptyset, \forall i, j \in [1, m], i \neq j, \text{and}$$

$$\mathfrak{G}_n^d = \bigcup_{i=1}^{m} C_i, i \in [1, m].$$

A partition $\mathfrak{C}$ of $\mathfrak{G}_n^d$ for which the above definition holds is called a *clustering*, and the subsets C_i are called clusters. The cluster problem then consists of finding a suitable clustering of $\mathfrak{G}_n^d$ satisfying a similarity criteria between genes in $\mathfrak{G}_n^d$, which is represented in most cases as a distance function ρ.

This clustering definition is very important since different clustering definitions could yield distinct results for the cluster properties in $\mathfrak{G}_n^d$. Note that in our case, a cluster C of a clustering in a metric space $\mathfrak{G}_n^d$ is a special type of subset in $\mathfrak{G}_n^d$ where no gene of C belongs to a different cluster of C in the same clustering.

From definitions C.8 and C.9 in Appendix C, we can prove that a cluster C of $\mathfrak{G}_n^d$ is a closed set (see Proposition D.2 in Appendix D). As a consequence of this result, C

is not an open set in $\mathfrak{G}_n^d$ and the frontier of C coincides with its boundary (Definitions C.7 and C.11, Appendix C). This proposition is very important since if C was an open cluster, then it makes no sense to think about its boundary points.

Hereinafter, to differentiate clusters in a clustering and clusters in a reference partition, we will call *reference clusters* (in shorthand, *r-clusters*) to the clusters of a reference partition.

7.2.2 Multidimensional Algorithm to Obtain Cluster Boundaries

The boundary points of a subset in a d-dimensional space are very important in Data Mining, since analyzing these points can reveal information about the problem being addressed [137, 250]. To compute the boundary of a cluster it is necessary to introduce the concept of an *extreme gene*, which is fundamental in searching for cluster boundary points. An i^{th} extreme gene (or an extreme gene) of a cluster is a gene (regarded as a vector in $\mathfrak{G}_n^d$) whose i^{th} component is either greater or less than for the remaining genes in the cluster. The set of all extreme genes of a cluster C is denoted by Exm$C$ (see Definition D.1 in Appendix D). An important result from the above statement is that an extreme gene of a cluster belongs to the boundary of such a cluster (as stated in Proposition D.3 in Appendix D). Note that if C is a cluster of $\mathfrak{G}_n^d$ then $\text{Exm}C \subseteq \text{Bd}C$ and $Card(\text{Exm}C) \leq 2d$, where Bd$C$ is the boundary of C and $Card$ is the cardinality of Exm$C$.

We now introduce an algorithm called *ClusterBoundary* to find the boundary points of a cluster. In contrast with other approaches, our boundary definition focuses on the concept of a boundary in a metric space, namely the set of points in the closure of a cluster that do not belong to the interior of the cluster:

7.2.3 *ClusterBoundary* Algorithm

Input: C, a cluster in $\mathfrak{G}_n^d$ and ρ the metric defined on $\mathfrak{G}_n^d$.
Output: BdC, the boundary of the cluster C

```
1. $\text{Bd}C = \emptyset$;
2. while $C \neq \emptyset$ do
```

```
3.      % Module (I) - Find all $i^{th}$ extreme genes in $C$.
4.      % max.exm, min.exm search for the maximal and minimal $i^{th}$ extreme genes
5.      % respectively.
6.      Exm$C = \emptyset$;
7.      **for all** $i \in [1, d]$ **and** $C \neq \emptyset$ **do**
8.          Exm$C :=$ Exm$C \bigcup \{g^i_{\succ} :=$ max.exm$(C, i)$, $g^i_{\prec} :=$ min.exm$(C \backslash \{g^i_{\succ}\}, i)\}$;
9.          $C := C \backslash \{g^i_{\succ}, g^i_{\prec}\}$;
10.     **endfor**
11.     Bd$C :=$ Bd$C \bigcup$ Exm$C$;
12.     % Module (II) - Compute the centroid (middle point) of Exm$C$.
13.     $a :=$ centroid(Exm$C$);
14.     % Module (III) - Compute the mid-points between extreme point pairs except
15.     % for $g^i_{\succ}$ and $g^j_{\prec}$ where $i = j$.
16.     $P_m := \emptyset$;
18.     **for all** $i \in [1, d]$ **do**
19.         $P_m := P_m \bigcup \{$middle.point$(g^i_{\succ}, g) \mid g \in$ Exm$C \backslash \{g^i_{\succ}, g^i_{\prec}\}\}$;
20.         $P_m := P_m \bigcup \{$middle.point$(g^i_{\prec}, g) \mid g \in$ Exm$C \backslash \{g^i_{\succ}, g^i_{\prec}\}\}$;
21.     **endfor**
22.     % Module (IV) - Compute the radius of a ball with interior points in $C$:
23.     choose either $r := \min\{\rho(a, p) \mid p \in P_m\}$ or $r :=$ mean$\{\rho(a, p) \mid p \in P_m\}$ or
24.     $r := \max\{\rho(a, p) \mid p \in P_m\}$;
25.     % choosing one of the above radiuses determines the type of approximation
26.     % to the boundary of the cluster.
27.     % Remove interior points of the ball with center $a$ and radius $r$.
28.     $C := C \backslash N(a, r)$;
29. **endwhile**
30. **end.**
```

Algorithm $ClusterBoundary$ is divided in four fundamental modules, which are explained using an example of a cluster in $\mathbb{R}^3$. Module (I) (lines 3-11) carries out a search for extreme points as shown in Figure 7.1-a. In this figure, the extreme points of a hypothetical cluster are highlighted in red. At each iteration of the algorithm, the cluster boundary is incrementally built from the extreme points (based on Proposition D.3 in Appendix D). Module (II) (lines 12-13) computes the centroid of the extreme points, which will be the centre of the interior point ball as shown in Figure 7.1-b. Note that the lines drawn between the extreme points form an eigth-sided polygon encloses most of the points. Module (III) (lines 14-21) computes the mid-points between each extreme point-pair, except for the pairs $(g^i_{\succ}, g^j_{\prec})$ where $i = j$. Figure 7.1-c shows this, as well as the possible radiuses computed from the centroid to the mid-points. Module (IV) (lines 22-28) determines the radius of the ball with the centre already computed in module (II). The radius can be chosen as either the minimal, the mean or the maximal distance between the centroid and the points in the set of mid-points P_m. The option of choosing different radiuses is related to the strategy for constructing the boundary of the cluster, which can be either conservative (minimal), intermediate (mean) or aggressive (maximal). This strategy depends on how different radiuses can be used to discriminate an increasing number of interior points. The ball (a sphere in $\mathbb{R}^3$) with the chosen radius is shown in Figure 7.1-d. All interior points of the ball, which are also interior points of the cluster, are removed (see Figure 7.1-e), whereas the polygon determined by the extreme points is considered to be the current approximation of the boundary.

Figure 7.1-f shows the result of the algorithm after one iteration, where convergence toward the cluster boundary can be viewed. The next iteration takes this new cluster as its input and the whole process is repeated until an empty cluster is obtained. If the size of the input cluster for this algorithm is less than $2d$, all the genes are then moved to the cluster boundary. A complete execution of the algorithm for a 2D cluster is given in Figure 7.2, where it has been shown the fast convergence of the algorithm towards the cluster boundary, which makes this algorithm suitable for an interactive environment. As shown in this figure, the algorithm runs three iterations, where the input cluster for each iteration is shown on views a1, a2 and a3. The process of computing the boundary points has been shown on b1 and b2. The boundary computed in each iteration is shown on c1, c2 and c3, where c3 is the output of the algorithm.

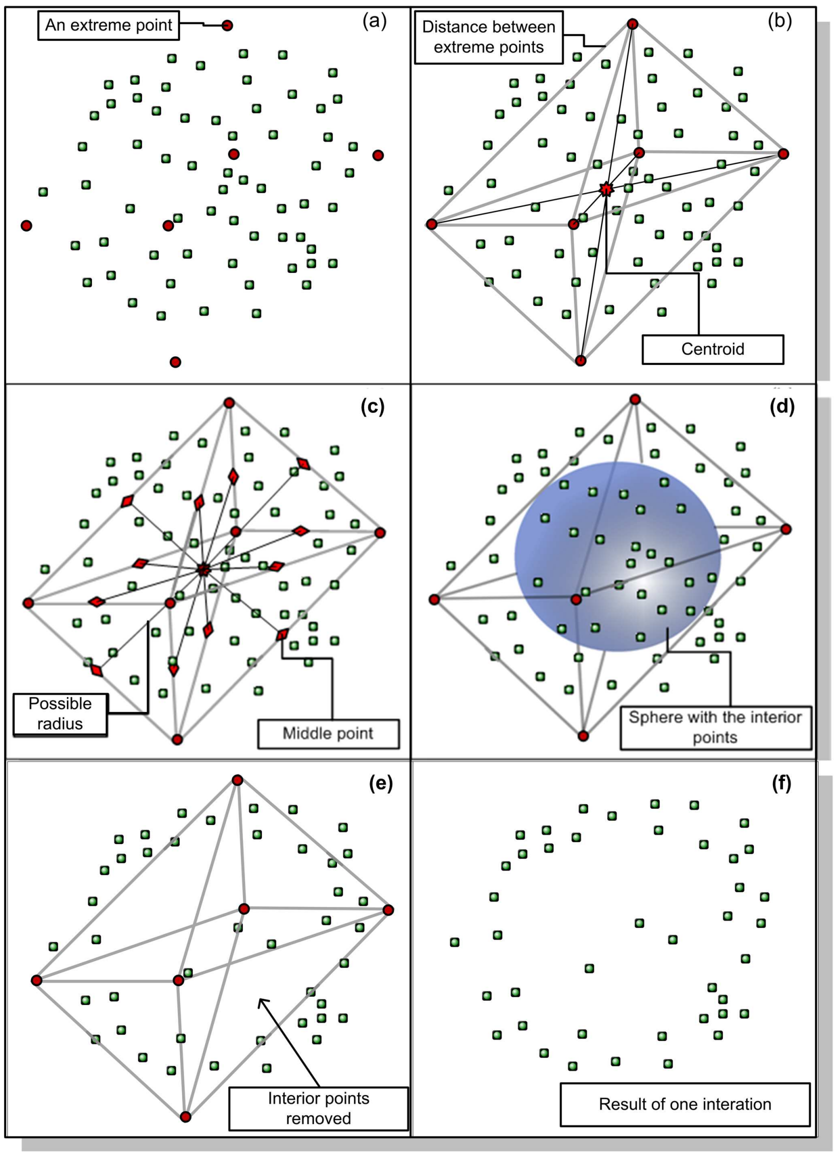


Figure 7.1: (a) A 3D gene-point cluster with its extreme points; (b) the centroid of the extreme points and the distances (shown as lines) between the extreme points; (c) the mid-points between the extreme points, and the radiuses computed from the centroid to the mid-points; (d) a sphere (ball) built from the centroid and a chosen radius in Figure c; (e) interior points removed from the sphere; (f) the extreme points from (e) are moved to the set of boundary points, and a new cluster arises.

As the input cluster to iteration 3 just has one gene-point, it has been moved to the boundary on c3 and the algorithm ends. Finally, the cluster with its boundary points and the shape of it have been shown on views d1 and d2 respectively. Note that the algorithm converges towards the boundary of the cluster in a few iterations.

According to the performance and the runtime complexity of *ClusterBoundary* [19, 97, 247], note that the number of mid-points in P_m is bounded by $2d(d-1)$ (see Proposition D.4 in Appendix D). Thus, we can conclude that the runtime of this algorithm in the worst case scenario is $O(k^2)$, where k is the size of the input cluster (for a formal proof see Proposition D.5 in Appendix D).

7.2.4 Surface Reconstruction based on Boundary Points

Surface reconstruction considers the extraction of shape information from a point set. These point sets often contain noise, redundancy and systematic variation arising from the experimental procedure. Therefore, a general approach to reconstructing surfaces is a challenging problem [119, 126].

The goal of surface reconstruction methods can be described as follows: *given a set of sample points X assumed to lie on or near an unknown surface U, construct a surface model S approximating U* [125, 127, 179].

The proposed algorithm is a modification of the one presented in [34, 37], which reconstructs convex hulls. Since many surfaces generally are not convex hulls, we provide a version that transforms the basis algorithm to obtain surfaces of non-convex hulls. As a general schema, our algorithm projects boundary points of a cluster onto the plane, and determines the convex boundary points. Second, the algorithm inserts the remaining non-convex boundary points into the list of convex boundary points. Finally, the algorithm returns an ordered list of boundary points which is used to establish the connectivity of points in a 3D dimensional space.

The first aim of this algorithm is to reconstruct cluster surfaces based on boundary points as a new alternative for cluster visualization. Thus, the input of this algorithm is the cluster boundary found by the *ClusterBoundary* algorithm. Note that this strategy improves the runtime of our algorithm for surface reconstruction, which is another advantage over the basis algorithm. The second aim is to represent the r-cluster surfaces (in a reference partition) of a data set in such a way that the genes in a cluster that belong to a r-cluster can be viewed within such a translucent r-cluster

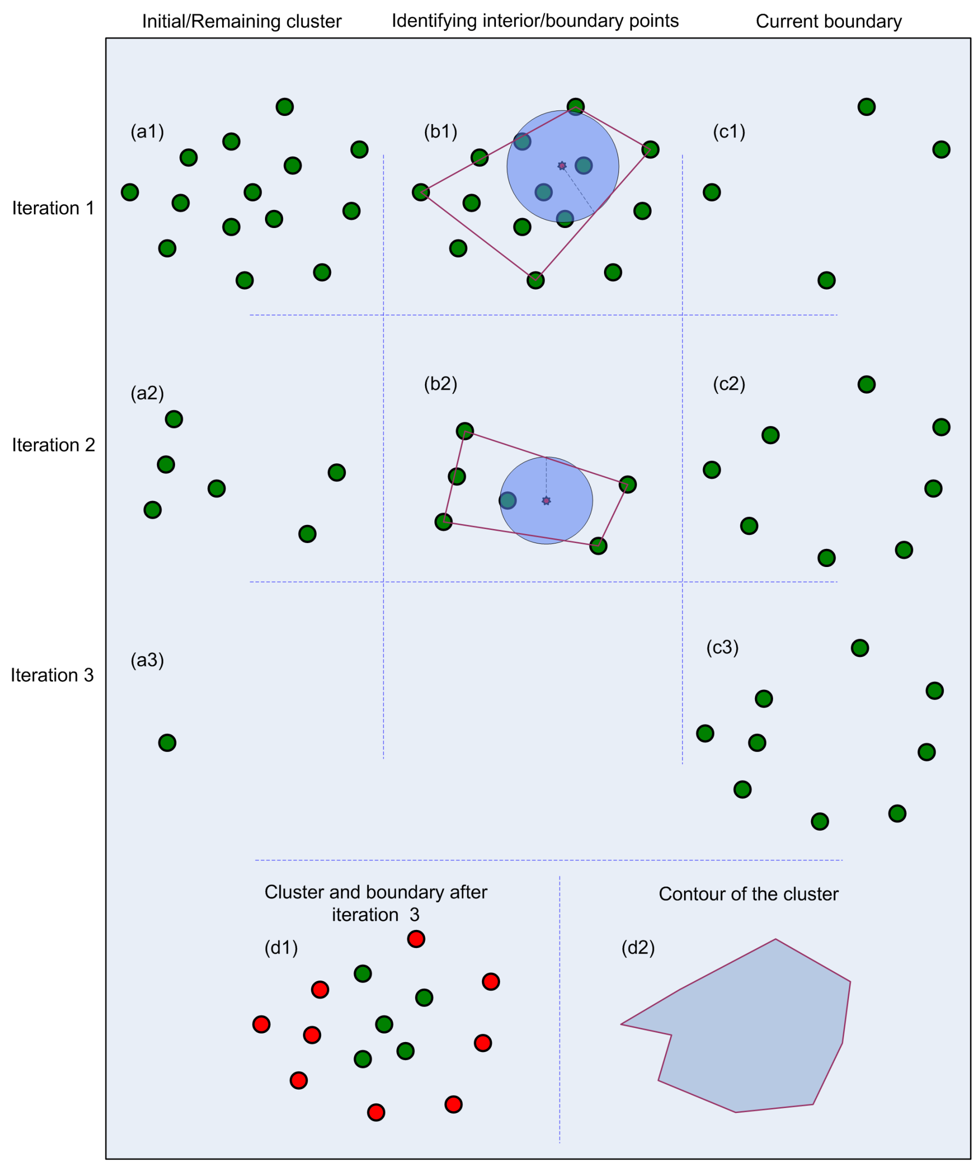


Figure 7.2: Steps of the *ClusterBoundary* Algorithm applied to cluster (a1). The performance of the algorithm using minimum radius is shown under three iterations. Intermediate computations are shown on (b1) and (b2). Remaining cluster and boundary of each iteration are respectively shown on {(a2), (a3)} and {(c1), (c2), (c3)}.

surface in a 3D space. The pseudocode for this algorithm, called *BoundaryShape* is outlined below.

7.2.5 *BoundaryShape* Algorithm

Input: B, the boundary of a cluster in $\mathfrak{G}_n^d$ and ρ the metric defined on $\mathfrak{G}_n^d$.

Output: boundary.idx, index sorted list of genes in B.

Require: DPoint function, a generic function to support different criteria for including non-convex points in the boundary.

```
1. $B'$:= Projection2D($B$); % Projects the genes of $B$ onto the plane.
2. $B'$:= Sort$_x$($B'$); % Sorts $B'$ by the $x$ coordinate.
3. fisrtpos := 1; % Position of the first gene in $B'$.
4. % Initialize the index lists of upper and lower points in $B'$.
5. lupper := llower := NULL;

6. % Find the position of upper convex points.
7. **while** there exists ($i >$ firstpos) so that UpperConvexEdge(firstpos, $i$, $B'$) holds **do**
8.     Add(lupper, $i$); % Adds $i$ to the end of lupper list.
9.     firstpos := $i$;
10. **endwhile**
11. % Find the position of the lower convex points.
12. fisrtpos := 1; Add(llower, fisrtpos);
13. **while** there exists ($i >$ firstpos) so that LowerConvexEdge(firstpos, $i$, $B'$) holds **do**
14.     Add(llower, $i$);
15.     firstpos := $i$;
16. **endwhile**
17. Remove the last index of lupper;
18. % Join both index lists of points, inverting lupper.
```

```
19. boundary.idx := Append(llower, Reverse(lupper));
20. % Take remaining non-convex point indexes out.
21. nc := NonconvexIdx(boundary.idx);
22. % Insert each index of nc in boundary.idx suitably.
23. for each $i \in$ nc do
24.     % Take the point-index of boundary.idx with the least distance to $i$ out.
25.     idx := $\arg_j \min\{\rho(i,j) | j \in$ boundary.idx$\}$
26.     % Determine whether $i$ is inserted to the left or to the right of idx in
27.     % boundary.idx. DPoint function returns a value based on a distance
28.     % criterion for three points.
29.     if DPoint($i$, idx, idx - 1) > DPoint($i$, idx, idx + 1) then
30.         Insert(boundary.idx, $i$, idx + 1)
31.     else
32.         Insert(boundary.idx, $i$, idx)
33.     endif
34. endfor
35. end.
```

The *BoundaryShape* algorithm is an incremental approach, which has three main parts. Namely, building a list of upper convex point indexes (lines 6-10), a list of lower convex point indexes (lines 11-16), and finally, to insert non-convex point-indexes into the convex indexes list (lines 17-33). So *BoundaryShape* carries first out some operations such as projecting points onto the $x : y$ plane and then sorts these points by x-coordinate in an ascendent way. Afterwards, it makes a search from left to right (as shown in Figure 7.3) for upper convex points and lower convex points (*while* loop, lines 7-10 and 13-16 respectively). Once found these points, they are added to *lupper* and *llower* lists respectively. The *UpperConvexEdge* and the *LowerConvexEdge* check whether the edge formed by the vertices *firstpos* and i is an upper or lower convex one,

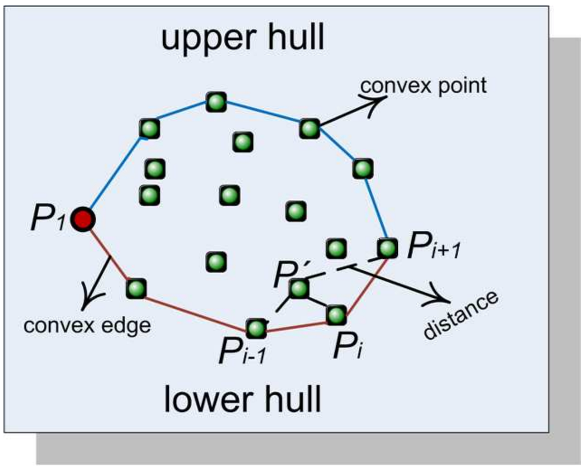


Figure 7.3: Convex hull of the boundary points of a cluster.

Figure 7.3. Therefore, the first *while* loop (lines 7-10) of the algorithm computes only those convex hull vertices that lie on the upper hull (blue edges in Figure 7.3). That is, the part of the convex hull running from the leftmost point P_1 to the rightmost point P_{i+1}, when the points are listed clockwise order as shown in Figure 7.3. The second *while* loop (lines 13-16) does the same but for vertices that lie on the lower hull (red edges).

At the end of the algorithm, *boundary.idx* is formed by joining *llower* and the reverse of *lupper*, which allows the convex points to be stored in anti-clockwise order. The set *nc* stores the remaining non-convex points that are finally inserted into *boundary.idx* in the *for* loop (lines 23-34). The value of *idx* in this loop is the point nearest to i (the convex point closest to P' in Figure 7.3 is P_i). We then have to decide whether i is inserted to the left or the right of *idx*, which is done using $DPoint$. $DPoint$ is a generic function that can use a number of distance criteriums as shown in Figure 7.3. Briefly, the point P' in Figure 7.3 can be inserted to the left or right of P_i based on the smaller of the distances between (P', P_{i-1}) and (P', P_{i+1}), the smaller of the distances from P' to the edges (P_i, P_{i-1}) and (P_i, P_{i+1}), the shorter of the paths connecting (P', P_{i-1}, P_i) and (P', P_{i+1}, P_i), or the smaller of the areas $\Delta\overline{P'P_{i-1}P_i}$ and $\Delta\overline{P'P_{i+1}P_i}$. Note that, in this figure, P' should be inserted to the left of P_i (the area of $\Delta\overline{P'P_{i-1}P_i}$ is less than the area of $\Delta\overline{P'P_{i+1}P_i}$), implying that the edge (P_i, P_{i-1}) is replaced by the edges (P', P_{i-1}) and (P', P_i). This mechanism allows a convex boundary to become non-convex (see Figure 7.2-d2).

The information given by the output of the *BoundaryShape* algorithm is used for the 3D triangulation of a surface that approximates the shape of the cluster whose boundary was the input to the algorithm.

7.3 The 3D-VC Framework

As a result of all previously presented on the visual framework, we have implemented tool 3D-VC (as a prototype), which is able to explore dendrograms, clusterings and clusters interactively with different views [13, 14]. This prototype uses *principal component analysis* (PCA, see Appendix B) to reduce data dimensionality to $\mathbb{R}^3$, so that a first approximation of data distribution can be analyzed on a 3D scatter plot. Furthermore, parallel coordinate visualization [15] and DNA microarray data views (heat map) using a color scale corresponding to gene expression levels are also presented.

7.3.1 Exploring the Tool

In order to explore different views of 3D-VC, the *Agnes* clustering method had been used to build the dendrogram (in the R language) that is displayed along with other views by the tool. Note that in this case, data and the algorithm have just been selected as an example to show the functionality of the tool, so the user can choose any other combination of data sets and algorithms to achieve its purpose. A general view of our prototype has been presented in Figure 7.4, which displays a sketch of eight linked views and six tasks of visual cluster analysis. The tasks proposed by 3D-VC in this figure, implicitly states a methodology to follow in the visual analysis and validation of the results of a clustering method. We then explain below, each task of this methodology to describe the 3D-VC tool.

Task 1 in Figure 7.4 has as its first goal to read the results of a clustering method applied to a data set of DNA microarray from an external source (in our case, from R language). As a result of the input, this task generates the HD-view, which shows (for this example) the *agnes* dendrogram with its microarray (as a heat map). This view provides an overall analysis of the whole clustering process. The second goal of Task 1

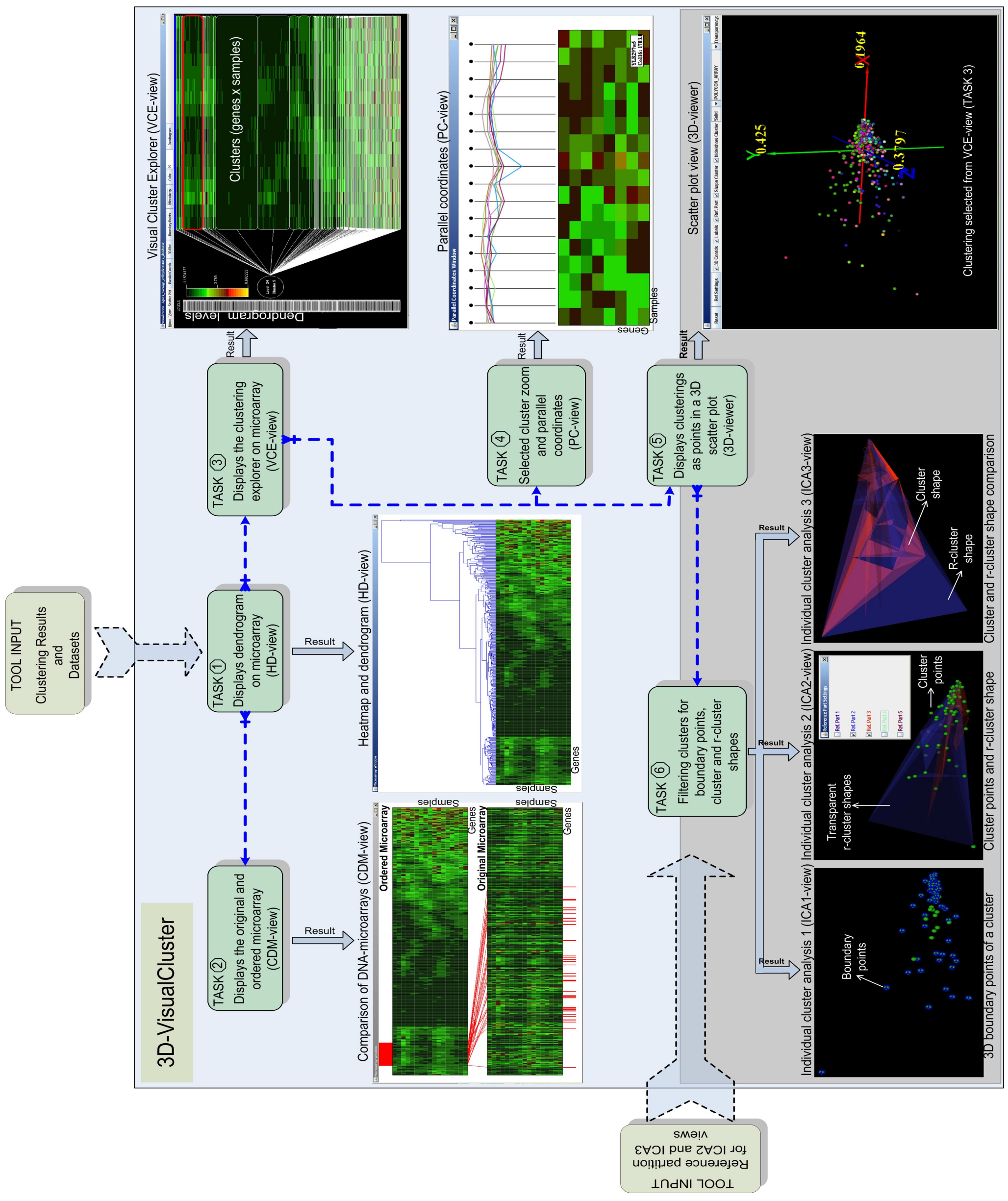

3D-VisualCluster
TOOL INPUT
Clustering Results and Datasets
TASK ① Displays dendrogram on microarray (HD-view)
TASK ② Displays the original and ordered microarray (CDM-view)
TASK ③ Displays the clustering explorer on microarray (VCE-view)
TASK ④ Selected cluster zoom and parallel coordinates (PC-view)
TASK ⑤ Displays clusterings as points in a 3D scatter plot (3D-viewer)
TASK ⑥ Filtering clusters for boundary points, cluster and r-cluster shapes
Result
Visual Cluster Explorer (VCE-view)
Clusters (genes x samples)
Dendrogram levels
Parallel coordinates (PC-view)
Genes
Samples
Scatter plot view (3D-viewer)
Clustering selected from VCE-view (TASK 3)
Heatmap and dendrogram (HD-view)
Comparison of DNA-microarrays (CDM-view)
Ordered Microarray
Original Microarray
Individual cluster analysis 1 (ICA1-view)
Individual cluster analysis 2 (ICA2-view)
Individual cluster analysis 3 (ICA3-view)
Boundary points
Transparent r-cluster shapes
Cluster points
Cluster shape
R-cluster shape
3D boundary points of a cluster
Cluster points and r-cluster shape
Cluster and r-cluster shape comparison
TOOL INPUT
Reference partition for ICA2 and ICA3 views

Figure 7.4: General view of the tool. There are eight linked views: microarray, dendrogram and parallel coordinates views at the top of the figure; 3D scatter plot views at the bottom; cluster boundary points, reference partition surfaces and cluster surface reconstruction.

is to generate from the input, two new tasks, 2 and 3, which give way to a new analysis on different views. For its part, Task 2 in Figure 7.4 has as a goal to compare the original and the ordered microarray according to the applied clustering method using the CDM-view. Genes in a cluster on the ordered microarray can also be located on the original microarray (through the red lines between both microarrays). This task is optional in the analysis process and is also useful to compare several clustering methods according to the reordering applied by them to the original microarray.

On the order hand, a continuation of Task 1 is Task 3, which has as a goal to locally explore the clusterings (and clusters) of the dendrogram shown by Task 1, through the result given by the VCE-view. Every level (clustering) of the dendrogram can be chosen on the left side in the VCE-view, and the clusters of the chosen level are shown on the right side. Therefore, this view explores in detail each cluster in the dendrogram and moreover, it is a new visual aid for exploring clusters on the microarray.

From Task 3, two possible tasks, Tasks 4 and 5, can be followed. Task 4 has as a goal to carry out a zoom-in of the selected cluster in the VCE-view and show it as a result by including parallel coordinates in the PC-view. Parallel coordinates are added for each sample of genes in the cluster, providing a means of comparison of the cluster quality. Task 5 has as a goal to give a different visualization alternative from the VCE-view. In this case, the clustering selected in the VCE-view (Task 3) has been shown in form of a 3D scatter plot by applying PCA (through the covariance matrix) to reduce the gene space dimensionality to obtain the view of the 3D-viewer. Each cluster in the current clustering has been displayed on the 3D-viewer in a different color. That is, gene-points in the same cluster have the same color while gene-points in different clusters have different colors. Each cluster on the 3D-viewer can be filtered for separate analysis.

Accordingly to the above, Task 5 generates the last task (task 6) of visual analysis, which has as a goal to filter clusters from the clustering represented on the 3D-viewer for their separately analysis through views ICA1, ICA2 and ICA3. The ICA1-view displays the currently selected cluster with its boundary gene-points, which have been computed by our algorithm *ClusterBoundary*. Note that before to analyze views ICA2 and ICA3, we need to again read from an external source to the tool, the reference partition of *cellcycle* as shown in Task 6 (Figure 7.4). Consequently, in the ICA2-view, we can choose each r-cluster 3D shape (built from our algorithm *BoundaryShape*) from the reference partition to visually compare it with the current cluster. This way, the r-cluster shape that better match (by visual inspection) the current cluster can be selected as shows the ICA2-view. Basically, to select the r-cluster shape most similar to the analyzed cluster (represented by the cloud of points), we observe the gene-points that fall within the r-cluster translucent shape, the ones on the border of the r-cluster shape and the ones outside of it. On the other hand, the ICA3-view is an alternative with respect to the ICA2-view, where the shape of the analyzed cluster is reconstructed (algorithm *BoundaryShape*) to be compared with the r-cluster shapes. This view provides another way of comparing a cluster with a r-cluster, which allows us to reinforce the assumptions taken into account in the ICA2-view. In our example, the current cluster as both gene-points (ICA2-view) and shape (ICA3-view) visually match the indicated r-cluster in both views, which means it is a good cluster according to the reference partition. Note that each task in Figure 7.4 provides different ways to see and analyze a cluster.

Representing reference partitions from r-cluster boundary points is one of the main contributions of this paper. This is because there are several statistical indicators for comparing a clustering with a reference partition, but there is no visual approach to validate this comparison. Each r-cluster in the reference partition is represented by our tool in a 3D surface. Thus, the gene-points at the intersection of a cluster with a r-cluster fall within the surface of such a r-cluster (in the space). In such case, one can check the degree of agreement between a clustering and a reference partition, or verify the results of statistical measures. Thus, it is possible to visually choose the level of the dendrogram that best approximates the reference partition.

Other filtering options from the scatter plot can be seen in Figure 7.5, where view a) shows the genes of the current cluster (selected on the VCE-view in Figure 7.4) in form of cubes. b) shows the same cluster but differentiated by a color different from the one of the remaining points, and c) isolates the cluster. Whereas three types of boundary corresponding to the maximum, mean and minimum radius are displayed in Figure 7.6 for the same cluster. Boundary points have been displayed in the form of cubes and have also been computed by algorithm *ClusterBoundary*. Note that the number of boundary points in Figure 7.6 increases from the maximum radius to minimum radius, as different radiuses imply different approximations of the cluster boundary. The radius determines the number of interior points to be removed from a cluster. Thus, the user can choose the type of boundary according to the application. That is, the maximum radius is more suitable to represent 3D surfaces of r-clusters, whereas a smaller radius may be more suitable to represent clusters by their boundary points (or their shapes).

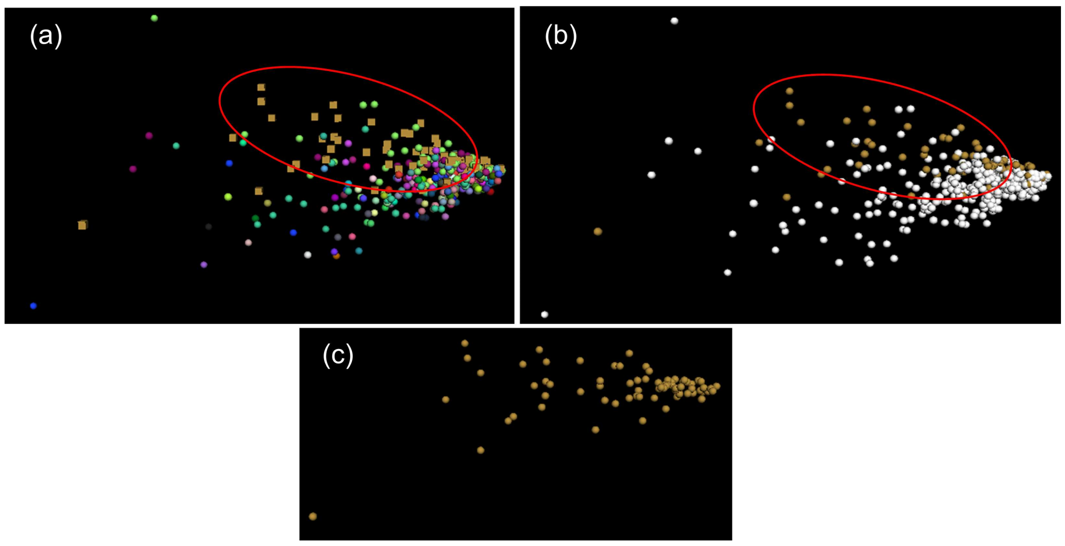


Figure 7.5: (a) compares the current cluster by changing the shape of the points (with cubes) vs. remaining points of the data set; (b) compares the current cluster (colored) vs. remaining points of the data set (white circle) and (c) displays only the current cluster.

7.3.2 Implementing the Framework

As stated earlier in this chapter and as a result of this section, algorithms *Cluster-Boundary* and *BoundaryShape* have been implemented in Java and Java 3D to build our visual framework through the 3D-VC tool. To complete the framework, it has also been implemented visualizations such as, dendrogram coupled with heat map of DNA microarray data, parallel coordinates coupled with clusters in form of heat map, clusters shown in form of gene-points or surfaces in a 3D space, and finally, comparison of clusterings with a reference partition of a data set. Each of these visualization components are linked together to create an interactive environment as discussed in comparative

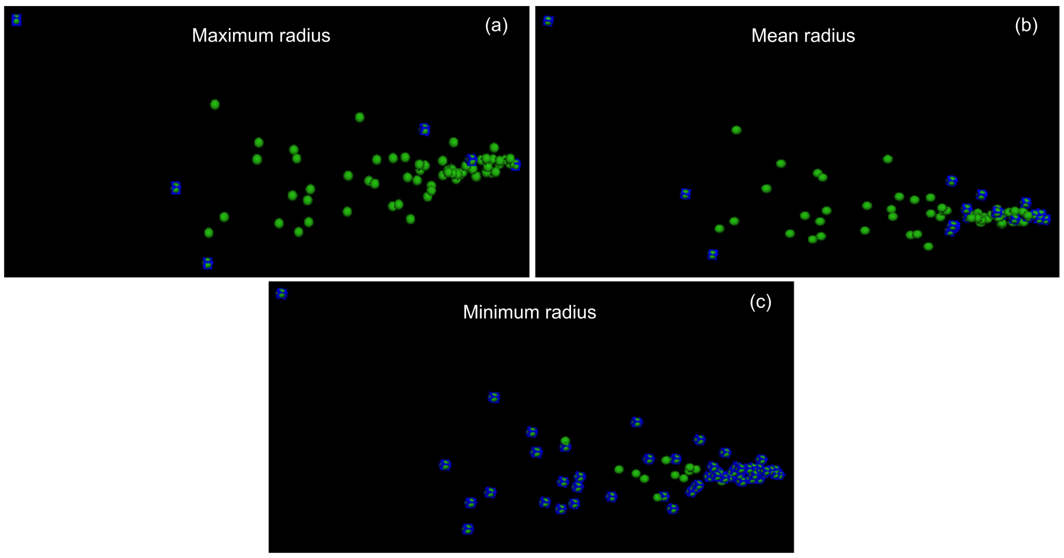


Figure 7.6: Shows boundary point options; (a) displays the boundary points computed from maximum radius; (b) boundary points computed from mean radius; and (c) boundary points computed from minimum radius.

Table 3.1, given in Chapter 3. Note that 3D-VC is linked to the R language, on which the clustering methods and the whole statistical analysis of the data are executed and the results are later read and displayed by the tool.

Although in the next chapter we prove the performance of our framework (through the 3D-VC tool) from a practical case, we have developed a set of images, videos, manuals and practical examples of the tool performance, also showing its reliability. All this additional material along with the implementation of the tool are publicly available at `http://www.analiticavisual.com/jcastellanos/3DVisualCluster/3D-VisualCluster`.

Chapter 8

Results from DNA Microarray Data

This chapter presents the results from evaluation studies of the proposed evolutionary method (EMHC) and framework (3D-VisualCluster or 3D-VC for short), both given in Chapters 6 and 7 respectively. To do that, on one hand, we study the behavior of EMHC on three public gene expression data sets and compare the results with other methods according to cluster validity measures as defined in Chapter 3. On the other hand, we also analyze the 3D-VC reliability for providing knowledge through its visualizations from the clustering results and the cluster validity measures applied to gene expression data. Additionally, the results of EMHC are also analyzed with visualizations of framework 3D-VC.

The chapter starts by detailing the scenario where EMHC and 3D-VC are tested, that is, stating data sets, hierarchical clustering methods and cluster validation measures to use. After that, firstly, EMHC is evaluated from a wide number of statistical tests and later, from visualizations given by 3D-VC. Secondly, the evaluation and analysis of framework 3D-VC are made on a practical case study. At the end of each section, we summarize the main results.

8.1 Used Data Sets

The first DNA microarray data set to use is a simulated one of *yeast cell cycle* (called *cellcycle*), original from [58], modified in [254] and published at `http://faculty.washington.edu/kayee/cluster`. The final gene expression matrix of *cellcycle* is composed of 384 genes evaluated under 17 conditions. *cellcycle* has been partitioned by [254] into 5 clusters of genes, which are assumed as a reference partition. *cellcycle* has been used as benchmark data to test clustering methods. As previously said in Chapter 7, clusters in a reference partition would be called r-clusters.

The second one is a real one called *sorlie*, which is composed of a gene expression matrix with 456 genes evaluated under 85 tissue samples. The original data had missing values scattered throughout the matrix. 10-nearest neighbors have been used for imputation as described in [54, 226].

The last one is also a real one called *lung*, which comprises 73 lung tissues including 67 lung tumors for 916 gene observations for each lung tissue, namely, a gene expression matrix of 916 genes $\times$ 73 samples. 20-nearest neighbors have been used to estimate missing values of this data set. The data set source at `http://genome-www.stanford.edu/lung_cancer/adeno/`, [99]. Finally, the three data sets have been standardized to mean 0 and variance 1.

8.2 Used Clustering Methods

We have focused the quality of clusters on terms of homogeneity (Homog), separation (Separ) defined in [137] and silhouette width (SilhoW, [10]) among other external measures given in Chapter 3. Five methods of hierarchical clustering are used to carry out comparisons: both *Agnes* (Euclidean distance between data and *mean-link* as distance between clusters) and *Diana* (Euclidean distance between data) in [10], *Eisen* (Euclidean distance between data) in [86], *HybridHclust* in [55] and *TSVQ* in [174]. All these methods have also been explained in Chapter 3.

8.3 Experimental Evaluation of the EMHC Evolutionary Model

As mentioned in Chapter 6, EMHC has been implemented on *R language* (R Development Core Team [16], using packages [56, 176]), and the experiments have been carried out on a $3GHz$ computer of $2GB$ of memory, using *Debian GNU/Linux* as an operating system[1].

The evaluation of EMHC has first been made on its genetic operators and afterwards, we have evaluated the goodness of the individuals generated by the evolutionary process of the method. After that, three sections have been given, one for each data set previously explained, dedicated to compare the results of EMHC with regard to the ones of the previously considered methods.

8.3.1 Evaluation of EMHC According to Genetic Operators

This subsection carries out validations of EMHC related to the performance of the first mutation operator (FMO) and the crossover operator (CO) defined by EMHC in Chapter 6, and the goodness of generated individuals according to the used fitness function. To do that, the experiments have been carried out on the *cellcycle* data set.

Mutation Operator Evaluation (FMO)

The goal of this experiment is to evaluate the performance of EMHC using only the FMO (without the crossover operator) and varying the values of mutation probability. This way, determining whether the FMO has a uniform behavior (on the fitness of the generated individuals) regardless of its mutation probability. The experiment has then been carried out taking into account the parameters in Table 8.1-(A), for which EMHC has been executed with 20 different settings, each using different mutation probability values with an increase of 0.05. For each probability value, the fitness value of the most fit individual in 100 generations has been taken out, as shows Figure 8.1. This figure displays four executions of the same experiment, where the curve in blue color represents the 20 fitness values reached by all tested settings for each execution,

[1]The R implementation of the method is available at `http://cran.r-project.org/web/packages/clustergas`.

whereas the curve in green color (along with the standard error bars) describes the average fitness computed from the fitness values in the four executions for each value of mutation probability.

Table 8.1: Parameter settings of EMHC for the evaluation of the genetic operators.

Parameter	(A) - FMO Evaluation	(B) - CO Evaluation
Crossover probability	0	$[0, 0.95]$
Mutation probability	$[0, 0.95]$	0
Number of individuals	2	2
δ	3/4	3/4
τ	$[0.15, 0.40]$	$[0.15, 0.40]$
ϵ	0.03	0.03
α	0.90	0.90
Metric on data	*Euclidean*	*Euclidean*

As Figure 8.1 shows, the reached fitness values remain above 35, most of them ranging between values 36 and 38 approximately (blue curve). Note that the average fitness curve (green curve) and the overlapping of the standard error bars show that the FMO has a uniform behavior bounded by fitness values between 36 and 38. This means that regardless of the mutation probability values, the FMO is able to improve the individuals by reaching fitness values higher than 35, whereas the individuals of the initial population have fitness values of about 30. All this shows the effectiveness of the operator.

Crossover Operator Evaluation (CO)

We now evaluate the behavior of the CO through EMHC with the same goal as the FMO. The EMHC parameter settings for this experiment is the same as the one used in the FMO experiment, but varying this time the crossover probability instead of the mutation probability, as shown in Table 8.1-(B). Four graphs of EMHC using only the CO (without the mutation operator) are shown in Figure 8.2. Each point on the blue curve of these graphs represents an EMHC regardless execution of 100 generations for the 20 tested values of crossover probability. The points on the green curve (along with

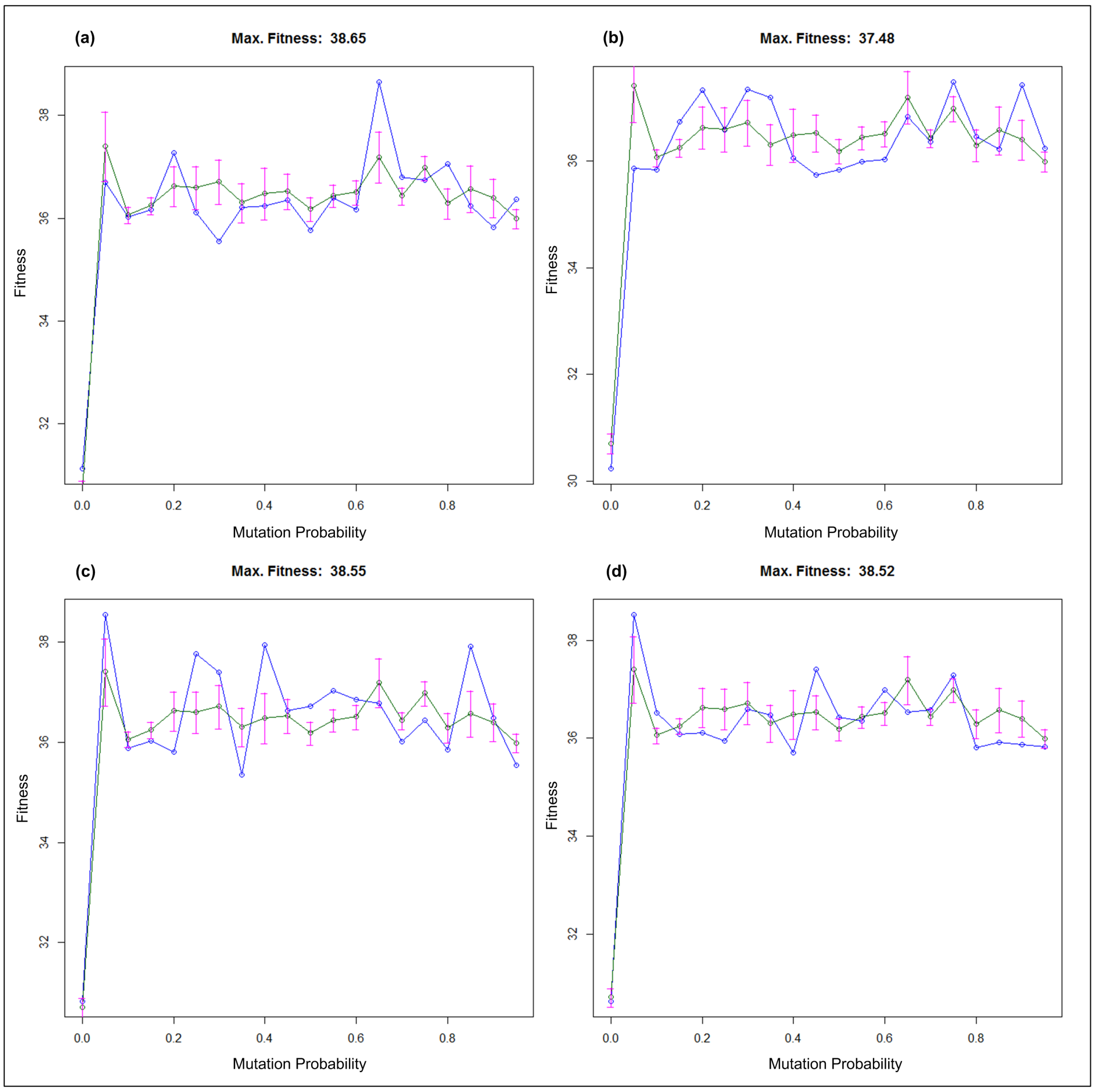


Figure 8.1: Four graphs of the same experiment, showing the fitness of the most fit individual in 100 generations of EMHC for 20 values of mutation probability (curve in blue color). The mean fitness from the four graphs (curve in green color) is shown along with its corresponding standard error bars.

the standard error bars) describe the mean fitness computed from the four executions of the same experiment.

As shows Figure 8.2, the results obtained for the FMO are also valid for the CO. Note that although the FMO has reached good fitness values, the best fitness value has been reached by the CO, that is, 39.19. This can be related to the neighborhood notion associated with the FMO, which allows us to find out new individuals, only within the neighborhoods of the individuals in the initial population (convergence toward local optimums). However, the CO explores other neighborhoods of solutions which are different from the ones of the recombined individuals. Therefore, the CO is potentially capable of finding better individuals than the FMO (possibility of finding a global optimum). In contrast, the convergence of the CO is slower than the FMO, precisely due to the CO looking for new neighborhoods in an extremely large search space, carrying out more meaningful jumps across the search space than the FMO, which searches only within local regions given by the mutated individuals.

Goodness of the Individuals

We are now interested in the evolution analysis between the individuals in the initial and final population of EMHC in terms of the global profile from the fitness values of each clustering in a dendrogram. The goal of this experiment is to verify the goodness of the individuals generated in the evolutionary process of EMHC with respect to the individuals in the initial population. But before dealing with that matter, a remark about of the methodology used to reduce the dendrogram length will be made. Namely, EMHC has been executed many times in order to check the number of clusters in the best clustering (according to the *ac* coefficient, see Definition 6.8 in Chapter 6) for the best dendrogram returned by each execution. This way, the dendrogram length (δ parameter, Definition 6.1 in Chapter 6) has been reduced taking into account the clustering-levels returned by the *ac* coefficient[2].

[2]Note that based on the graphs in Figure 8.4, the current dendrogram length ($\delta = 3/4$) can be reduced as an alternative to the strategy described in this paragraph. For example, examining the profiles in Figure 8.4 we have that approximately from 60 clusters, the curve described by the clustering fitness is almost monotonic decreasing (in mathematical terms). Thereby, it is possible to remove that part of the dendrograms whose clusterings have more than 60 clusters and thus, optimizing the performance of the method. To do this, we rely on Proposition 6.1 and Definition 6.1 in Chapter 6: given that we want to remove all clusterings from 61 clusters ($k = 61$) and the current dendrogram length is $|\mathfrak{G}| = 384 - 2 - \lfloor 384 \cdot 3/4 \rfloor = 94$, then the level corresponding to 61 clusters is computed from

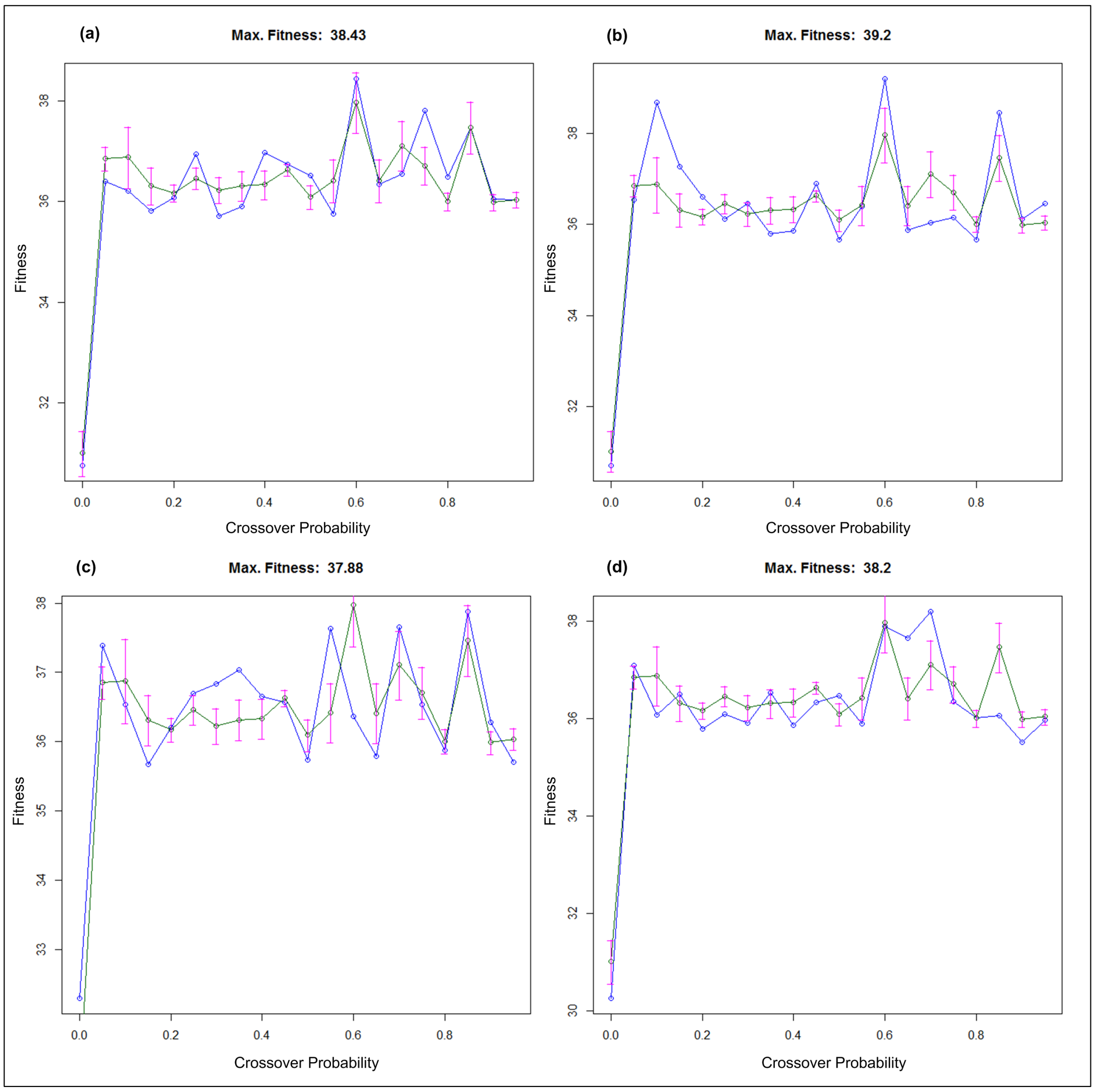


Figure 8.2: Four graphs of the same experiment, showing the fitness of the most fit individual in 100 generations of EMHC for 20 values of crossover probability (curve in blue color). The mean fitness from the four graphs (curve in green color) is shown along with its corresponding standard error bars.

To then measure the quality of the individuals, the profile of fitness values for all nested clusterings encoded by a given dendrogram is represented in a graph which plots the number of clusters in each clustering ($x - axis$) vs. the fitness values of each one ($y - axis$). The profiles from four random dendrograms in an initial population are shown in Figure 8.3. As shown in this figure, the curve described by these dendrograms presents much oscillations and there are many big differences between the fitness values of two consecutive clusterings (i.e., two adjacent and nested clusterings in the same dendrogram). Let us now see the evolution (improvement) of these individuals after the execution of EMHC with the parameters in Table 8.2.

Table 8.2: Parameter settings to evaluate the individuals of EMHC.

Parameter	Value (or interval)
Crossover probability	$[0.60, 0.75]$
Mutation probability	$[0.10, 0.20]$
Number of individuals	4
Generation number	1000
δ	$3/4$
τ	$[0.15, 0.40]$
ϵ	0.03
α	0.90
Metric on data	*Euclidean*

The fitness profile of the four dendrograms in the final population of EMHC are shown in Figure 8.4. As shown, the global fitness behavior of the dendrograms has been improved, since the profile curves of the final solutions (after applying the genetic operators of EMHC) are more monotonic and smoothed than the initial ones. Moreover, the maximum fitness value is reached when the number of clusters is between 30 and 50. If the above stands for many dendrograms, then the optimum number of clusters for the analyzed data set may be in this interval. Note that the best fitness values

$level = |\mathfrak{G}| - k + 2 = 94 - 61 + 2 = 35$. So, the new dendrogram length is $|\mathfrak{G}| - level = 94 - 35 = 59$ and finally, the δ that we need is computed from $\delta = \frac{n - 2 - |\mathfrak{G}|}{n} = \frac{384 - 2 - 59}{384} = \frac{323}{384}$. With this new δ it is possible to reduce the dendrogram length from 94 levels ($\delta = 3/4$) to only 59 levels. □

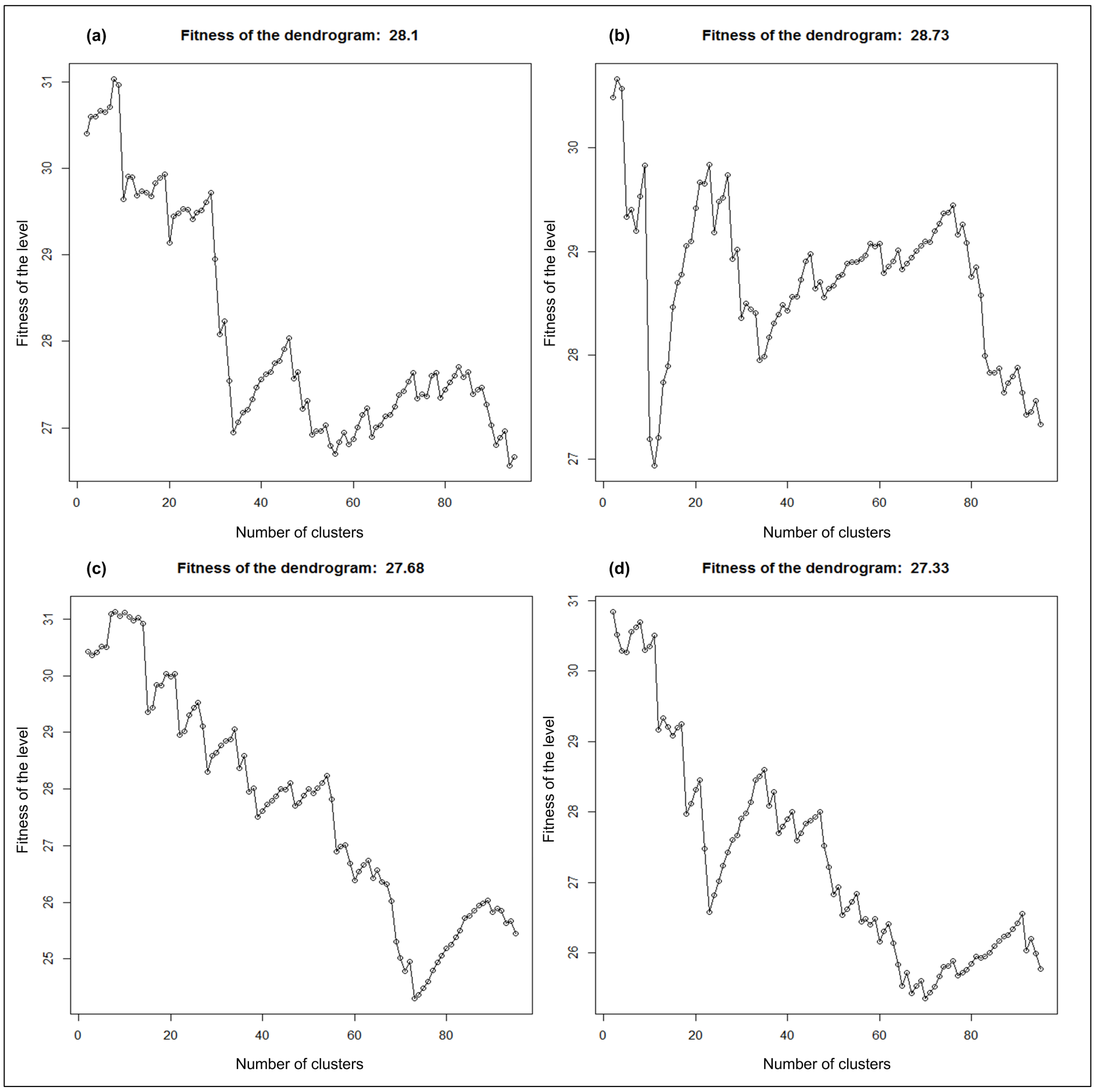


Figure 8.3: Fitness profile for four dendrograms of an initial population from the *cell-cycle* data set, showing the fitness values for each clustering of these dendrograms.

are achieved by the individuals of this figure and additionally, the clusterings (levels) located around a number of clusters between 30 and 50 are equivalent since they have similar fitness.

Observe that the curves described in Figure 8.4 tend to decrease from approximately 50 clusters, thereby, as far as the used data set is concerned, when the number of clusters increases, the quality of the clustering decreases. On the other hand, the quality of the dendrograms in Figures 8.3 and 8.4 is directly proportional to the area under the curves represented by them. That is, individuals of higher quality imply higher area under the curve. According to this, note that almost all clustering fitness values of the dendrograms in Figure 8.3 are below 30, while almost all ones in Figure 8.4 are above 30.

8.3.2 EMHC Evaluation from cellcycle

In what follows, a cluster validity process is performed to compare the results of EMHC. Therefore, we have focused on the quality of the clusters in terms of homogeneity (Homog), separation (Separ), silhouette width (SilhoW) and agreement with the reference partition of the data set. EMHC will be compared with *Agnes, Diana, Eisen, Hybrid-Hclust* and *TSVQ* as previously described in section 8.2.

Note that the measure values of homogeneity decrease when the clustering quality (or the dendrogram quality) increases, whereas the measure values of separation and silhouette width increase when the clustering quality (or the dendrogram quality) increases.

Homogeneity and Separation

The goal of this experiment is to show the performance of EMHC (on data set *cellcycle*) with respect to other methods by means of internal cluster validity measures. For this experiment, EMHC has been initialized and executed based on Table 8.3-(A), and its result has been given as input to Algorithm 6.1 defined in Chapter 6. The final output has then been extracted to make comparisons such as listed in Table 8.4, where EMHC is compared with the five preceding methods for the three validity measures. The values in Table 8.4 represent the mean of the validity measure values applied to each clustering of the output dendrogram for each method. In other words, the result of the validity

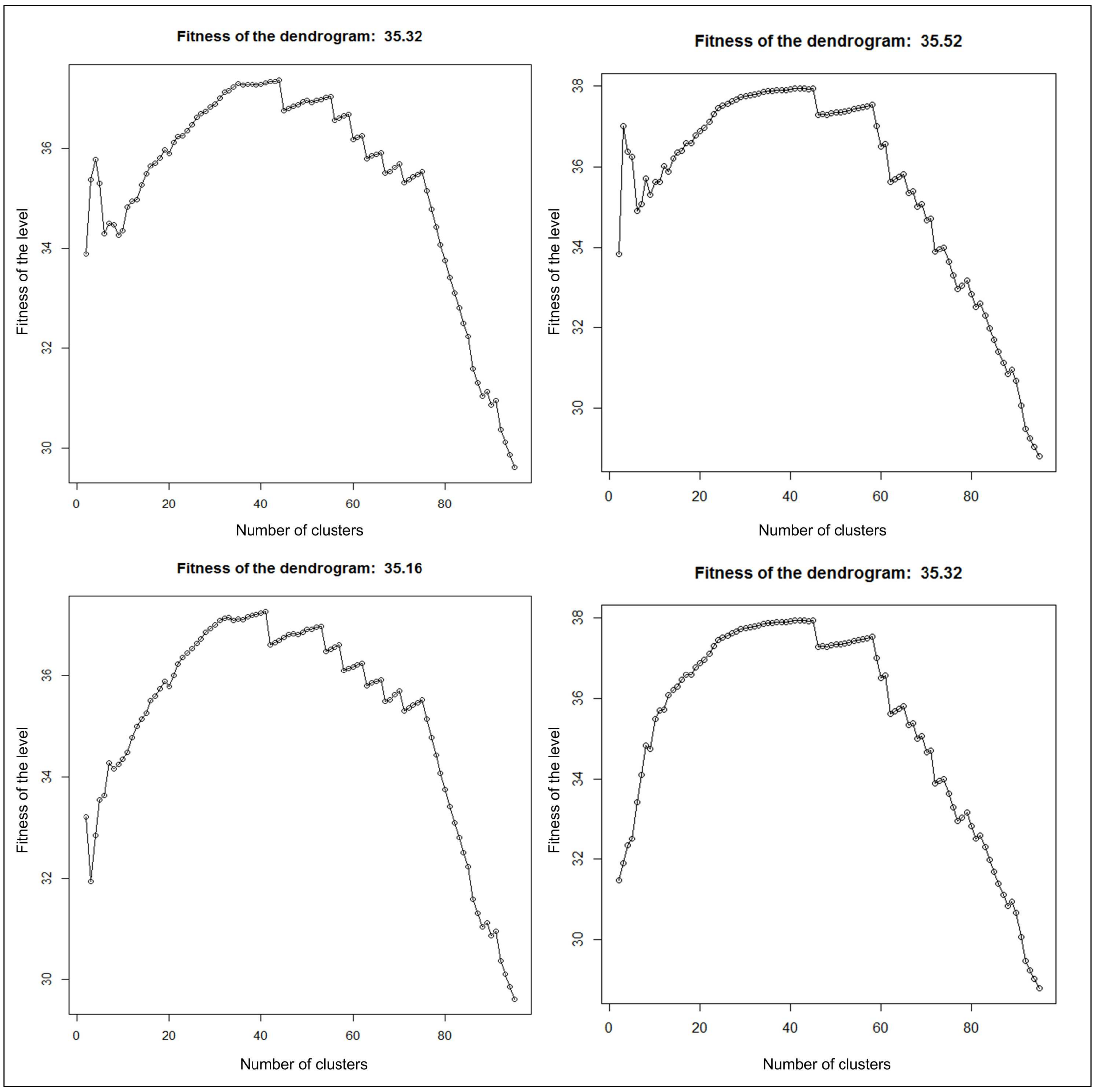


Figure 8.4: Fitness profile for four output dendrograms of EMHC from the *cellcycle* data set, showing the fitness values for each clustering of these dendrograms.

measures applied to the dendrograms returned by the methods. Best values for each measure have been underlined.

Table 8.3: Parameter settings to evaluate EMHC from the *cellcycle, sorlie* and *lung* data set.

Parameter	(A) - cellcycle	(B) - sorlie	(C) - lung
Crossover probability	$[0.60, 0.75]$	$[0.60, 0.75]$	$[0.60, 0.75]$
Mutation probability	$[0.10, 0.20]$	$[0.10, 0.20]$	$[0.10, 0.20]$
Number of individuals	10	10	20
Generation number	$[10^3, 10^5]$	$[10^3, 10^5]$	$[10^3, 10^6]$
δ	12/13	20/21	28/29
τ	$[0.15, 0.40]$	$[0.15, 0.40]$	$[0.15, 0.40]$
ϵ	0.03	0.03	0.03
α	0.90	0.90	0.90
Metric on data	*Euclidean*	*Euclidean*	*Euclidean*

Table 8.4: Cluster validity of EMHC vs. five hierarchical clustering methods on the *cellcycle* data set.

Method	Homog	Separ	SilhoW
Agnes	2.92	10.18	0.33
Diana	2.59	8.82	0.26
Eisen	4.21	14.50	0.46
HybridHclust	$\underline{2.22}$	7.87	0.24
TSVQ	2.24	7.84	0.24
EMHC	5.48	$\underline{18.99}$	$\underline{0.47}$

Table 8.4 shows methods EMHC and *Eisen* have achieved the best values with respect to separation and silhouette width on their dendrograms. However, the remaining methods perform better on homogeneity.

Agreement with the Reference Partition

The goal of this experiment is to show the performance of EMHC (on *cellcycle*) with respect to other methods by means of external cluster validity measures. To do this, we have carried out a comparison of the *cellcycle* reference partition (5 gene r-clusters[3]) with respect to each clustering with 5 clusters of the dendrograms given by the methods in Table 8.4. Firstly, the comparisons are made according to the cluster definition with the given validity measures. Table 8.5 lists the values given by the clusterings with 5 clusters of each method and the reference partition (assuming the reference partition as the output of a hypothetical algorithm).

Table 8.5: Comparison of six clustering methods vs. the reference partition of the *cellcycle* data set.

Method	Homog	Separ	SilhoW
Ref. Partition	6.13	6.60	-0.06
Agnes	3.86	12.57	0.55
Diana	3.94	12.87	0.49
Eisen	6.00	25.02	0.67
HybridHclust	<u>2.73</u>	8.70	0.33
TSVQ	2.83	8.67	0.32
EMHC	6.09	<u>25.73</u>	<u>0.69</u>

As shown in Table 8.5, the best values of separation and silhouette width have been reached over again by EMHC and *Eisen*. The remaining methods perform better for homogeneity. Moreover, all methods have performed better than the reference partition on the validity measures. Thus, on the basis of the cluster definition, the above results imply this reference partition can be improved.

Secondly, we now compare the agreement and disagreement degree of the clusterings in Table 8.5 with respect to the reference partition. Jaccard coefficient (JC) and Minkowski measure (MM) have been considered for this purpose. JC measures the extent of agreement of a clustering with regard to a reference partition and MM illustrates the proportion of disagreement as explained in Chapter 3. JC and MM

[3]The clusters in a reference partition are called *r-clusters*, see Chapter 7.

have been chosen for this experiment because they may be more effective than other measures in gene-based clustering [111, 137].

Table 8.6 compares each method with the introduced reference partition by means of the JC and MM measures. From this table we can emphasize that EMHC and *Eisen* (as in Table 8.4 and 8.5 for column Separ and SilhoW) have reached the best results on JC. In contrast, *HybridHclust* has achieved the best value for MM. The above results imply that there are many gene pairs in the EMHC and *Eisen* clustering that agree with the ones of the reference partition, more than the other methods, but these clusterings have not been partitioned in the same way as the reference partition.

Table 8.6: Cluster validity of six clustering methods with respect to the reference partition of the *cellcycle* data set.

Method	JC	MM
Agnes	0.20	1.62
Diana	0.21	1.63
Eisen	0.23	1.82
HybridHclust	0.17	$\underline{1.35}$
TSVQ	0.15	1.37
EMHC	$\underline{0.25}$	1.79

In addition to comparative Tables 8.5 and 8.6, an alternative view of them has been built in Figure 8.5. This figure represents the values of these tables in form of curve by means of a system of *parallel coordinates* ([15, 36, 96, 139, 246]) based on the used measures. The curves correspond to each method and describe the behavior of each one with respect to the validity measures. Consequently, in this figure there clearly are three groupings of similar behavior methods, namely $\{EMHC, Eisen\}$, $\{Agnes, Diana\}$ and $\{HybridHclust, TSVQ\}$.

8.3.3 EMHC Evaluation from sorlie

we now apply homogeneity and separation validation to the output of each clustering method for the *sorlie* data set as made for the *cellcycle* data set in the above subsection.

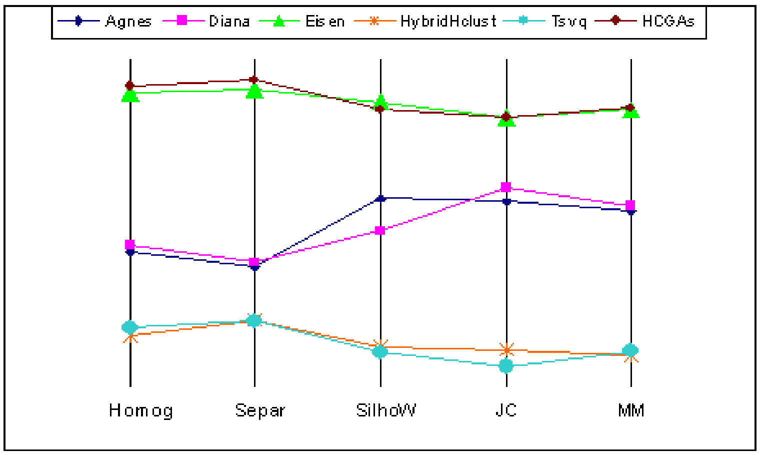


Figure 8.5: A view of parallel coordinates of five measures, representing six curves formed by the values reached from each method in tables 8.5 and 8.6 on the *cellcycle* data set.

Homogeneity and Separation

The goal of this experiment is to show the performance of EMHC (on *sorlie*) with respect to other methods by means of internal cluster validity measures. In this experiment EMHC has been parameterized and executed based on Table 8.3-(B), and its result has also been given as input to Algorithm 6.1. Table 8.7 lists the values reached by each validity measure applied to each dendrogram such as in Table 8.4.

Table 8.7: Cluster validity of EMHC vs. five hierarchical clustering methods on the *sorlie* data set.

Method	Homog	Separ	SilhoW
Agnes	16.84	24.77	<u>0.20</u>
Diana	15.83	18.26	0.05
Eisen	16.24	17.58	-0.01
HybridHclust	15.26	17.56	0.03
TSVQ	<u>15.23</u>	17.58	0.03
EMHC	16.89	<u>25.32</u>	<u>0.20</u>

Additionally, Table 8.8 shows the validity values applied to the best clustering (according to the *ac* coefficient, see Definition 6.8 in Chapter 6) of each output dendrogram in Table 8.7. The #Cluster column shows the number of clusters of each clustering chosen by *ac*.

Table 8.8: Cluster validity of EMHC vs. five hierarchical clustering methods based on the best clustering of each output dendrogram on the *sorlie* data set.

Method	#Cluster	Homog	Separar	SilhoW
Agnes	10	16.90	<u>24.67</u>	<u>0.19</u>
Diana	18	15.35	18.23	0.05
Eisen	5	16.32	17.64	0.00
HybridHclust	21	14.46	17.46	0.01
TSVQ	21	<u>14.27</u>	17.52	0.02
EMHC	16	16.69	24.27	0.14

Table 8.7 shows methods EMHC and *Agnes* have reached the best values with respect to separation and silhouette width. Table 8.8 has also scored the best results with respect to separation and silhouette width on the best clusterings for methods EMHC and *Agnes*, although for this case, EMHC does not perform better than *Agnes* for the same indicators. But the next subsection shows how these results of EMHC can be improved. The remaining methods of this table have been performed better for homogeneity. Furthermore, the values in the #Cluster column could be employed to estimate the optimum number of clusters.

An alternative view to Table 8.8 is presented in Figure 8.6. This graph represents the values of the above table in form of curves as made in Figure 8.5 (but for the *sorlie* data set). Note that unlike of Figure 8.5, there now are only two groupings with methods of similar behavior, $\{EMHC, Agnes\}$ and $\{HybridHclust, TSVQ\}$.

Improving Solutions of other Methods

In this subsubsection we show how the outputs of other clustering methods can be improved through EMHC. Since EMHC can take as input a population of individuals formed by the outputs of other methods, it can achieve better solutions than the ones

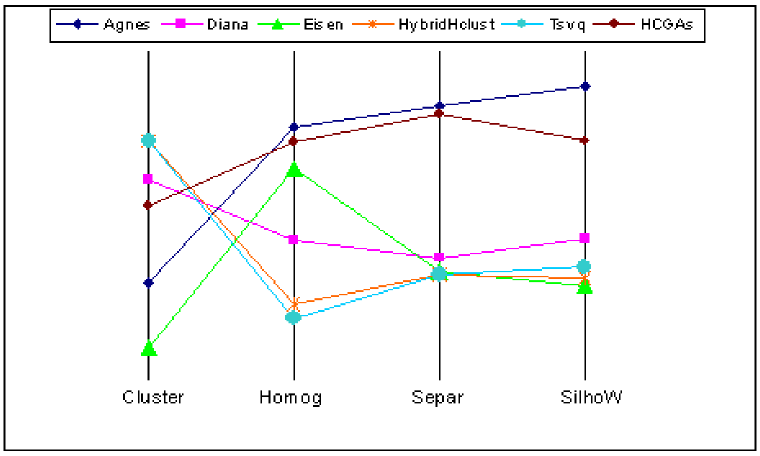


Figure 8.6: A view of parallel coordinates for four measures, representing six curves of the values reached by each method in Table 8.8 on the *sorlie* data set.

found by such methods. Besides that, due to most existing methods have a general propose on their performance, we can recombine their solutions through EMHC by defining a fitness function adapted to the given problem and suitably, fitting the EMHC parameters. It is then possible to obtain more fit solutions to the problem. This EMHC property, its ability of adapting solutions of other methods to the context of a determined problem is one of the most important contributions of our approach. Thus, the goal of this experiment is to improve the performance of EMHC by taking as initial population of individuals, the solutions given by the other methods.

On this basis, the five outputs of the other methods in Table 8.7 came to form the initial population for EMHC according to the parameter setting given in Table 8.3-(B). Two experiments has been carried out, one considering fitness function g_d (g_d is based on g_c, which looks for the criteria of homogeneity and separation at a time. See Definition 6.10 in Chapter 6) and the another one, using fitness function g_{rd} (see Definition 6.14 in Chapter 6), which adds a new objective to g_d to control the number of genes in each cluster as explained in section 6.3, Chapter 6. Both experiments are listed in Table 8.9 under the used validation measures.

On row g_d of case (A) in Table 8.9 (for the case of the measures applied to the whole dendrogram), we have that the values are not better than the ones of *Agnes* and

Table 8.9: Cluster validity of EMHC based on the solutions of other methods on the *sorlie* data set.

	(A)-Validity on the whole dendrogram			(B)-Validity of the best clustering			
Function	Homog	Separ	SilhoW	#Cluster	Homog	Separ	SilhoW
g_d	16.90	24.09	0.15	4	17.18	24.83	0.23
g_{rd}	15.20	17.57	0.04	21	14.29	17.50	0.01

EMHC in Table 8.7, although these values are better than the ones of the remaining methods (on the Separ and SilhoW column). This fact implies an improvement in the solutions of four methods in Table 8.7. Later, we will see (by 3D-VC) that from the visual point of view, such solutions in Table 8.9 can really be an improvement of the ones given by EMHC and *Agnes* in Table 8.7.

On row g_d of case (B) in Table 8.9 (it is the best clustering selected according to the *ac* coefficient, see Definition 6.8 in Chapter 6), EMHC has scored better values for separation and silhouette width than *Agnes* and itself in Table 8.8. As shown on such a row, the idea of improving solutions given by other methods (external solutions) through EMHC performs well. Even better, some of the results achieved have been better than before.

On the other hand, on row g_{rd} in Table 8.9, we can not expect the results be as good as g_d (on separation and silhouette width), because g_{rd} has one objective more than g_d. However, it has reached a meaningful improvement on homogeneity as can be verified from Tables 8.7 and 8.8. Note that for this case, EMHC has converged towards *TSVQ* since EMHC (according to g_{rd}) along with *TSVQ* scored the best homogeneity values with respect to all tables in this subsection. This shows that g_{rd} which introduces a new objective (according to parameter $\hat{t}$, Chapter 6) is able to improve the homogeneity values of the dendrograms guided by g_d, and that the users can select g_d or g_{rd} according to the application level. Note also that unlike g_d, g_{rd} sacrifices inter-cluster separation to gain intra-cluster homogeneity.

8.3.4 EMHC Evaluation from lung

We are now going to analyze the *lung* data set on the homogeneity and separation criteria such as in previous subsections. EMHC has been setup as in Table 8.3-(C) and its solution has been given to Algorithm 6.1.

Homogeneity and Separation

The goal of this experiment is to show the performance of EMHC (on *lung*) with respect to other methods by means of internal cluster validity measures. Then, Table 8.10 lists the values scored by each validity measure applied to the dendrograms of each method.

Table 8.10: Cluster validity of EMHC vs. five hierarchical clustering methods on the *lung* data set.

Method	Homog	Separ	SilhoW
Agnes	14.25	20.84	<u>0.14</u>
Diana	12.79	15.31	0.05
Eisen	13.27	15.17	-0.03
HybridHclust	12.15	15.03	0.05
TSVQ	<u>12.13</u>	15.03	0.04
EMHC	14.38	<u>21.60</u>	<u>0.14</u>

Table 8.11 shows the measure values applied to the best clustering (according to the *ac* coefficient, see Definition 6.8 in Chapter 6) of each output dendrogram in Table 8.10. Column #Cluster shows the number of clusters in each chosen clustering.

As shown in these two tables, the best values for separation and silhouette width have been achieved by EMHC and *Agnes* method. The best values for homogeneity have been achieved by methods *TSVQ* and *HybridHclust*. Figure 8.7 shows the above by representing each method as a curve, where the methods in groupings $\{EMHC, Agnes\}$ and $\{HybridHclus, TSVQ\}$ have similar behavior as happened for the *sorlie* data set.

Table 8.11: Cluster validity of EMHC vs. five hierarchical clustering methods based on the best clustering of each output dendrogram on the *lung* data set.

Method	#Cluster	Homog	Separ	SilhoW
Agnes	7	14.45	22.30	0.23
Diana	29	11.98	15.10	0.05
Eisen	9	13.40	15.19	-0.01
HybridHclust	31	<u>11.27</u>	14.90	0.03
TSVQ	31	11.31	14.90	0.03
EMHC	3	14.61	<u>23.11</u>	<u>0.30</u>

8.4 Experimental Evaluation of the 3D-VC Framework

This section uses the *cellcycle* data set to show the usefulness of 3D-VC in the context of knowledge discovery and visual analytics. Note that 3D-VC can be used as visual validation of clustering results, being an alternative to the existing statistical validity measures [67, 115, 137, 256]. As explained in Chapter 7, 3D-VC includes visualizations of microarrays, dendrograms, parallel coordinates and different views on a 3D scatter plot that some of them have been shown in this section.

8.4.1 A Case Study

This subsection presents a case study on the *cellcycle* data set and its reference partition with 5 r-clusters of genes. The goal of this case study is to show that the visual validation and the numerical validation (cluster validity measures) of the results from the clustering methods are consistent with respect to a reference partition of the given data set. Note that not always visual validation matches the numerical validation, since the numerical validations are based on different assumptions and in some cases, these ones do not represent the reality of a given problem. Then, to reach the previous goal, we first compare the results of three clustering methods with the reference partition for *cellcycle* by means of statistical indices that measure the similarity degree. Afterwards, we also visually compare the quality of the used statistical indices and show the relationships (numerical and visual) between the reference partition and the clustering results.

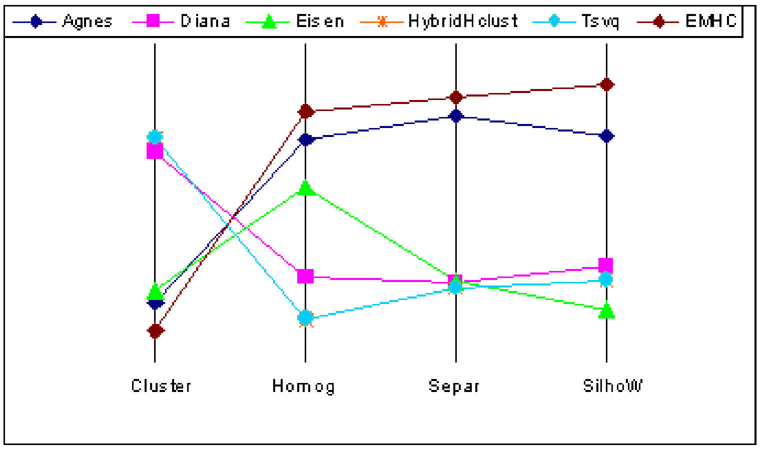


Figure 8.7: A view of parallel coordinates for four measures, representing six curves with values scored by each method in Table 8.11 (*lung* data set).

To this end, the hierarchical clustering methods *Agnes, Diana* and *Eisen* are applied to *cellcycle*. The following statistical indices are used: the Rand index (RI), the Jaccard coefficient (JC), the Minkowski measure (MM) [111, 225], all given in Chapter 3. JC and RI measure the extent of agreement between two clusterings, whereas MM illustrates the proportion of disagreement. When the agreement measure value increases, agreement grows and when the MM value decreases, disagreement becomes smaller. All these numerical indices provide evidence that strengthen or weaken the agreement between a clustering and a reference partition, which agrees with the goal of this case study.

Two cases (A and B in Table 8.12) representing different levels (clusterings) of the output dendrograms have been chosen to compare each method with the reference partition. Case A (column Case) selects the clustering with 5 gene clusters (column #Clusters) for each dendrogram, that is, the case where the number of clusters in the dendrograms matches that of the reference partition. Case B selects the clustering with 13 gene clusters for each dendrogram, namely, the clustering that represents a good structural grouping according to color intensity levels in the microarray representation (for example, Figure 8.8). The Method column of the table represents the name of the method applied in each case, while the remaining columns contain the values reached

Table 8.12: Comparative table of agreement and disagreement of clustering results with respect to the reference partition of *cellcycle*.

			Agreement		Disagreement
Method	Case	#Clusters	JC	RI	MM
Agnes			0.20615	<u>0.39175</u>	<u>1.63176</u>
Diana	A	5	0.20957	0.38518	1.64056
Eisen			<u>0.22690</u>	0.24387	1.81936
Agnes			0.16953	0.56706	1.37668
Diana	B	13	0.13083	<u>0.65473</u>	<u>1.22939</u>
Eisen			<u>0.20719</u>	0.39583	1.62622

for each index in each case. The best values for each index column are underlined.

Note that for Case A in Table 8.12, *Agnes* attains the best values for the RI and MM indices, while *Eisen* performs better for the index JC. This shows that *Agnes* has more indices with the best values than the other methods, giving further support to the conclusion that it fits better the reference partition. In Case B, *Diana* reaches the best values for the RI and MM indices, and *Eisen* for JC. Overall, except for JC, Case B has better index values than Case A. Thus, the gene distributions with 13 clusters fits better the reference partition than these of 5 clusters. Finally, since *Diana* in Case B has the best values of the whole table for RI and MM, we can conclude that its result fits better the *cellcycle* reference partition than the other methods.

Visual Check of the Results

As the final part of the case study, we now check that the index results for Case B correspond to the visual representation of the microarray view and with the reference partition of *cellcycle* in the scatter plot view. This allows us a validation process of all the used indices. Firstly, we show the microarray and dendrogram view for each method in Case B in Table 8.12 by Figures 8.8, 8.9 and 8.10, which verify that *Diana* finds a better gene cluster distribution than the other methods. That is, by a visual comparison of the clusters with similar colors, marked on the heat map (red rectangles) in those figures (8.8, 8.9 and 8.10), we can see *Diana* has better division of clusters than *Agnes* and *Eisen*. In contrast, *Eisen* (Figure 8.10) shows the worst division of clusters on the heat map.

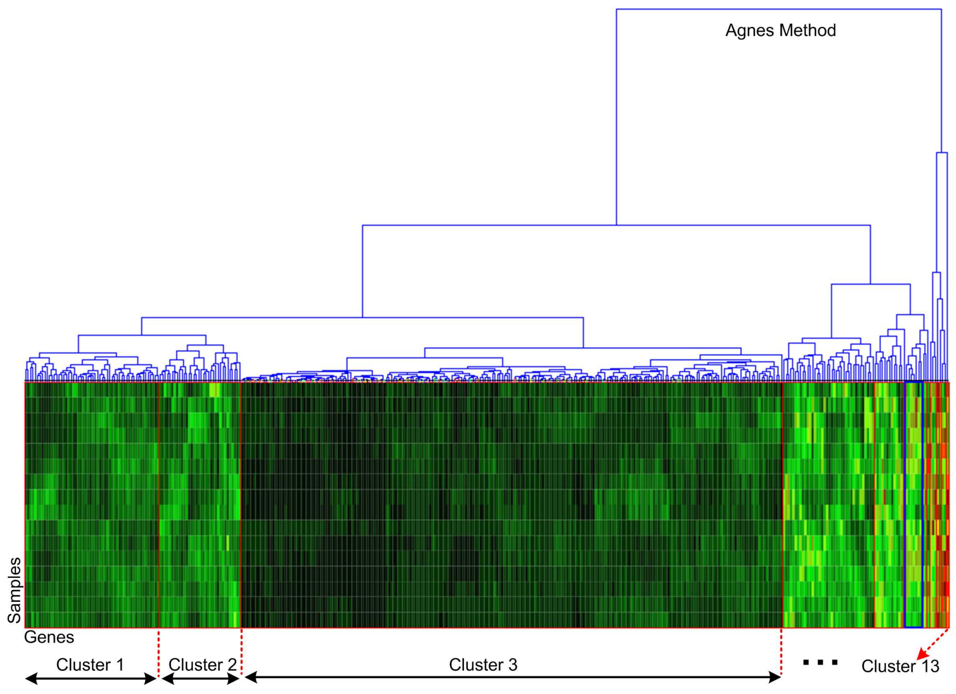


Figure 8.8: Dendrogram and microarray of *cellcycle* for *Agnes* method, showing the level of 13 clusters. Clusters are enumerated from left to right.

Secondly, we visually compare the clustering of each method in Case B against the reference partition, aimed at finding the method that yields the clustering most in agreement with the reference partition. A visual representation of the *cellcycle* reference partition is given by Figure 8.11, which shows the 3D-surfaces of its five gene r-clusters, computed from the ClusterShape algorithm. In order to compare the r-clusters with the clusters in Case B in Table 8.12, a visual inspection has been carried out for the clusters of each method similar to these r-cluster shapes.

Figures 8.12, 8.13 and 8.14 show the r-clusters (in the form of 3D-surfaces) overlapped on the clusters (represented as a cloud of gene-points) most similar to them for each method of Case B in Table 8.12. For *Agnes* in Figure 8.12, six visual matches have visually been found for only four clusters with respect to the reference partition. The remaining clusters for *Agnes* are not considered because they are very small and so

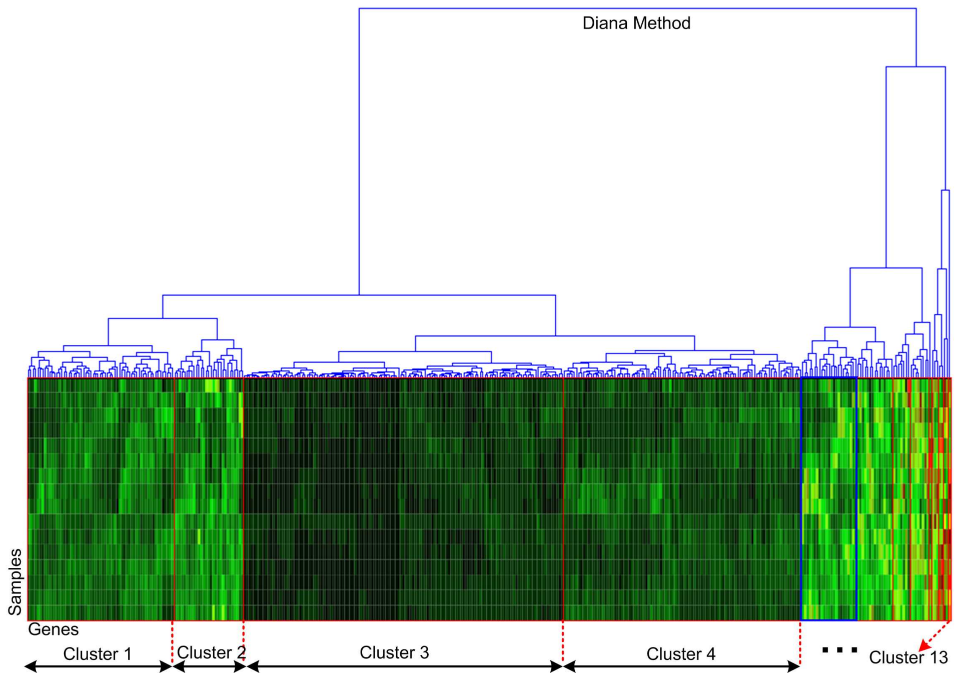


Figure 8.9: Dendrogram and microarray of *cellcycle* for the *Diana* method, showing the level of 13 clusters.

match any r-cluster. The process followed to choose the r-cluster shape most similar to a cluster is to first select each r-cluster shape on the cluster and visually analyze that the solid shape of the r-cluster matches the shape of the cloud of gene-points described by the cluster. In addition, we visually check the position of the gene-points of the cluster on the 3D-surface of the r-cluster as explained for ICA2-view in Figure 7.4. For *Diana* (Figure 8.13), seven matches have been found for only six clusters with respect to the reference partition. For the remaining clusters no matches have been found or the clusters have been very small. For *Eisen* (Figure 8.14) six matches have been found for only two clusters. Note that cluster 2 in this figure is repeated four times for different r-clusters. This is because cluster 2 is big and therefore, matches any r-cluster, as also happens for small clusters.

To summarize, we have completed three types of validation for the results of

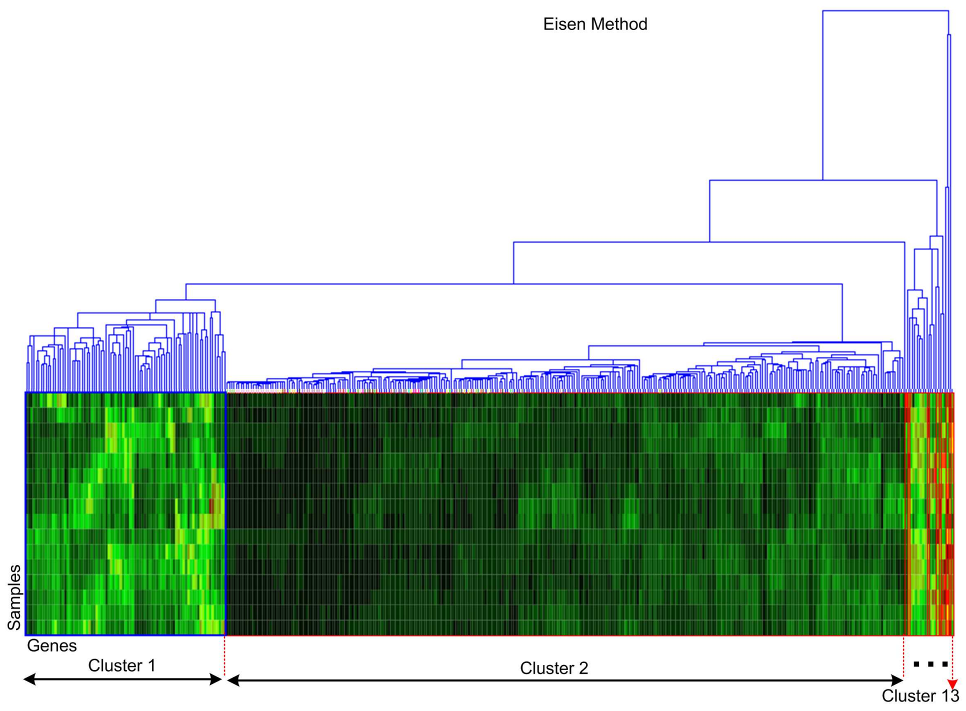


Figure 8.10: Dendrogram and microarray of *cellcycle* for *Eisen* method, showing the level of 13 clusters.

Agnes, Diana and *Eisen*. That is, we have validated the results using the indices in Table 8.12, using the dendrogram and microarray view in Figures 8.8-8.10, and finally by visual comparison with a reference partition in Figures 8.12-8.14. For this experiment, the three tests have given the same results. Namely, in Case B, *Diana* (Table 8.12) returned the best values for the used indices. Furthermore, *Diana* has shown the best distribution of gene clusters with respect to the microarray and dendrogram visualization, as displayed in Figure 8.9, and have also given the best matches (six matches) between the selected clustering and the reference partition for *cellcycle*, Figure 8.13. On the other hand, *Eisen* achieved the worst results for the three tests (the worst, in the sense that *Eisen* has been less similar to the reference partition than the other methods). In conclusion, we have validated that the *Diana* method has the best performance on *cellcycle* according to its reference partition, and consequently, the clustering that better represent *cellcycle* is that with 13 clusters given by *Diana*. Moreover, the

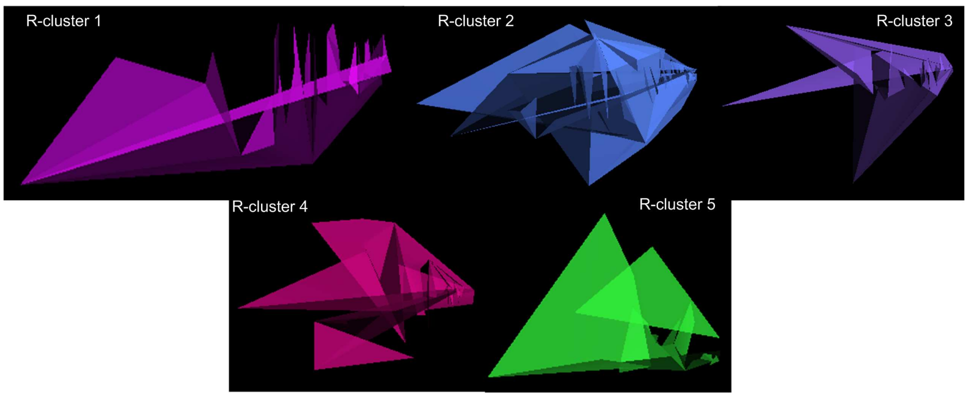


Figure 8.11: *cellcycle* reference partition. Each r-cluster has been displayed as a 3D surface.

most meaningful clusters in the above clustering have been identified according to the reference partition. These clusters are $\{1, 2, 3, 4, 5, 6\}$, as shown in Figures 8.9 and 8.13. We have that the quantitative and visual analysis agree and additionally, the visual analysis allows us to identify the clusters that match the r-clusters. All this shows the importance of the 3D-VC tool in cluster analysis of DNA microarray data[4].

8.5 Visual Analysis of the Results of the EMHC Evolutionary Model

At this point, we can display the results given in the tables of Subsections 8.3.3 and 8.3.4. We focus on the *sorlie* and *lung* data set since they have not a reference partition. The results obtained by the methods used on them have been displayed with the 3D-VisualCluster tool. This allows us to visually verify the quality of the returned dendrograms and extract conclusions on the performance of the used methods, since there is not a reference partition to externally validate the results. To do that, we have chosen the two clustering methods that have better performance on each of these data sets. That is, for each data set, the method that better has performed on homogeneity and the one on separation are selected to display their dendrograms. For both data

[4]Additional material on the tool performance at `http://www.analiticavisual.com/jcastellanos/3DVisualCluster/3D-VisualCluster`

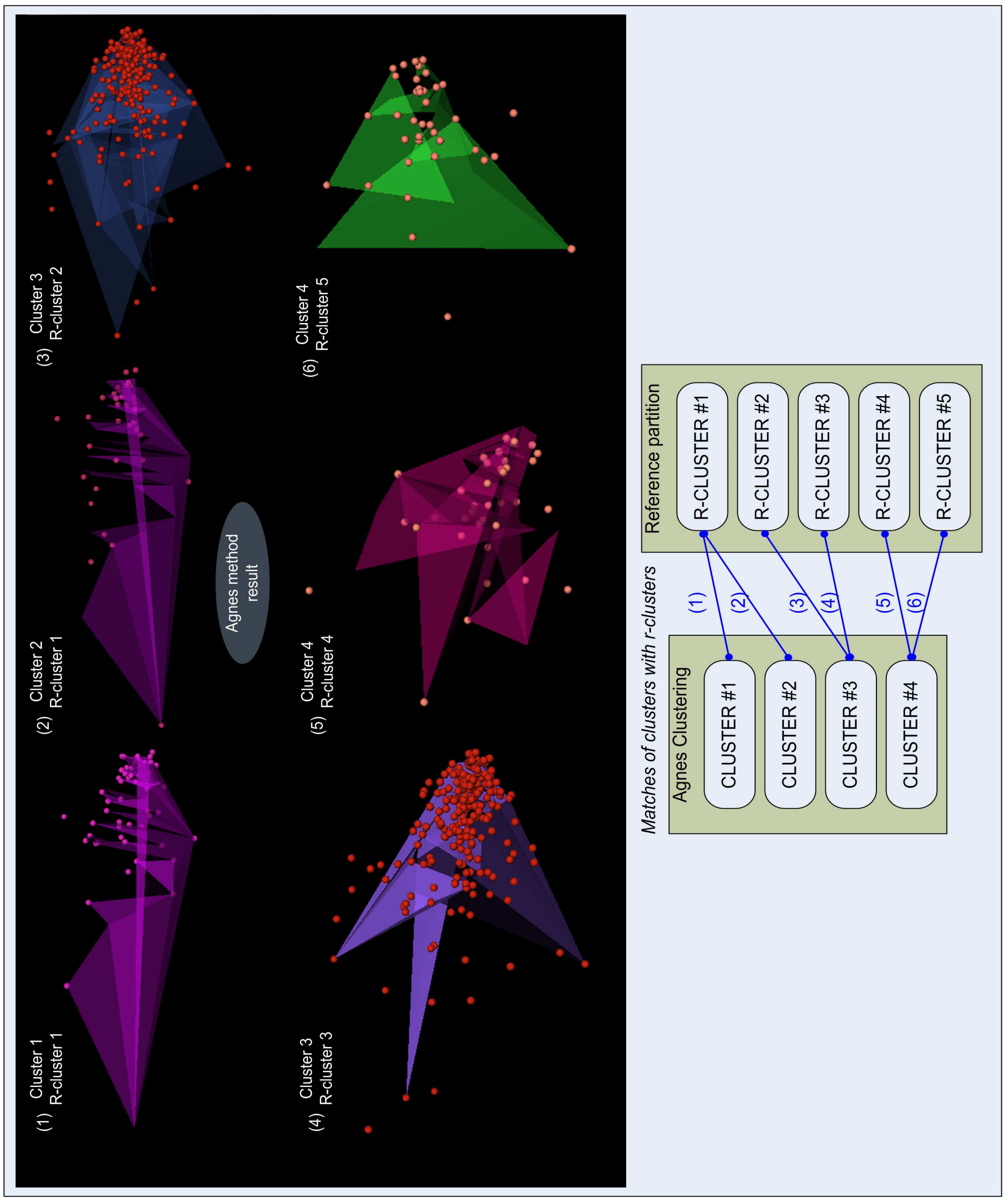

(1) Cluster 1 R-cluster 1
(2) Cluster 2 R-cluster 1
(3) Cluster 3 R-cluster 2
(4) Cluster 3 R-cluster 3
(5) Cluster 4 R-cluster 4
(6) Cluster 4 R-cluster 5
Agnes method result
Matches of clusters with r-clusters
Agnes Clustering
CLUSTER #1
CLUSTER #2
CLUSTER #3
CLUSTER #4
Reference partition
R-CLUSTER #1
R-CLUSTER #2
R-CLUSTER #3
R-CLUSTER #4
R-CLUSTER #5
(1)
(2)
(3)
(4)
(5)
(6)

Figure 8.12: Shapes of the *cellcycle* reference partition and the clusters that better match each r-cluster, *Agnes* method.

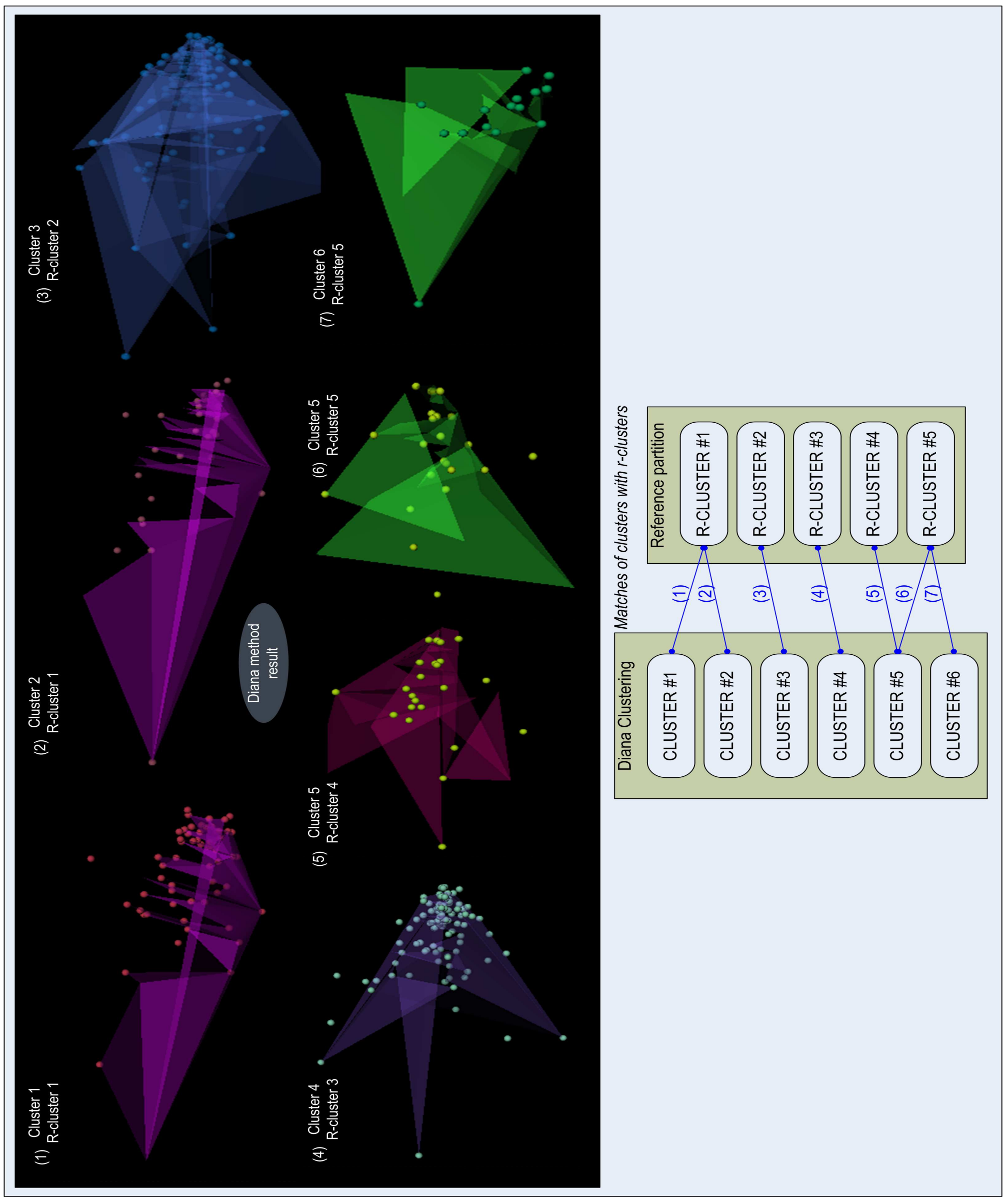

(1) Cluster 1 R-cluster 1
(2) Cluster 2 R-cluster 1
(3) Cluster 3 R-cluster 2
(4) Cluster 4 R-cluster 3
(5) Cluster 5 R-cluster 4
(6) Cluster 5 R-cluster 5
(7) Cluster 6 R-cluster 5
Diana method result
Matches of clusters with r-clusters
Reference partition
R-CLUSTER #1
R-CLUSTER #2
R-CLUSTER #3
R-CLUSTER #4
R-CLUSTER #5
Diana Clustering
CLUSTER #1
CLUSTER #2
CLUSTER #3
CLUSTER #4
CLUSTER #5
CLUSTER #6
(1)
(2)
(3)
(4)
(5)
(6)
(7)

Figure 8.13: Shapes of the *cellcycle* reference partition and the clusters that best match each r-cluster, *Diana* method.

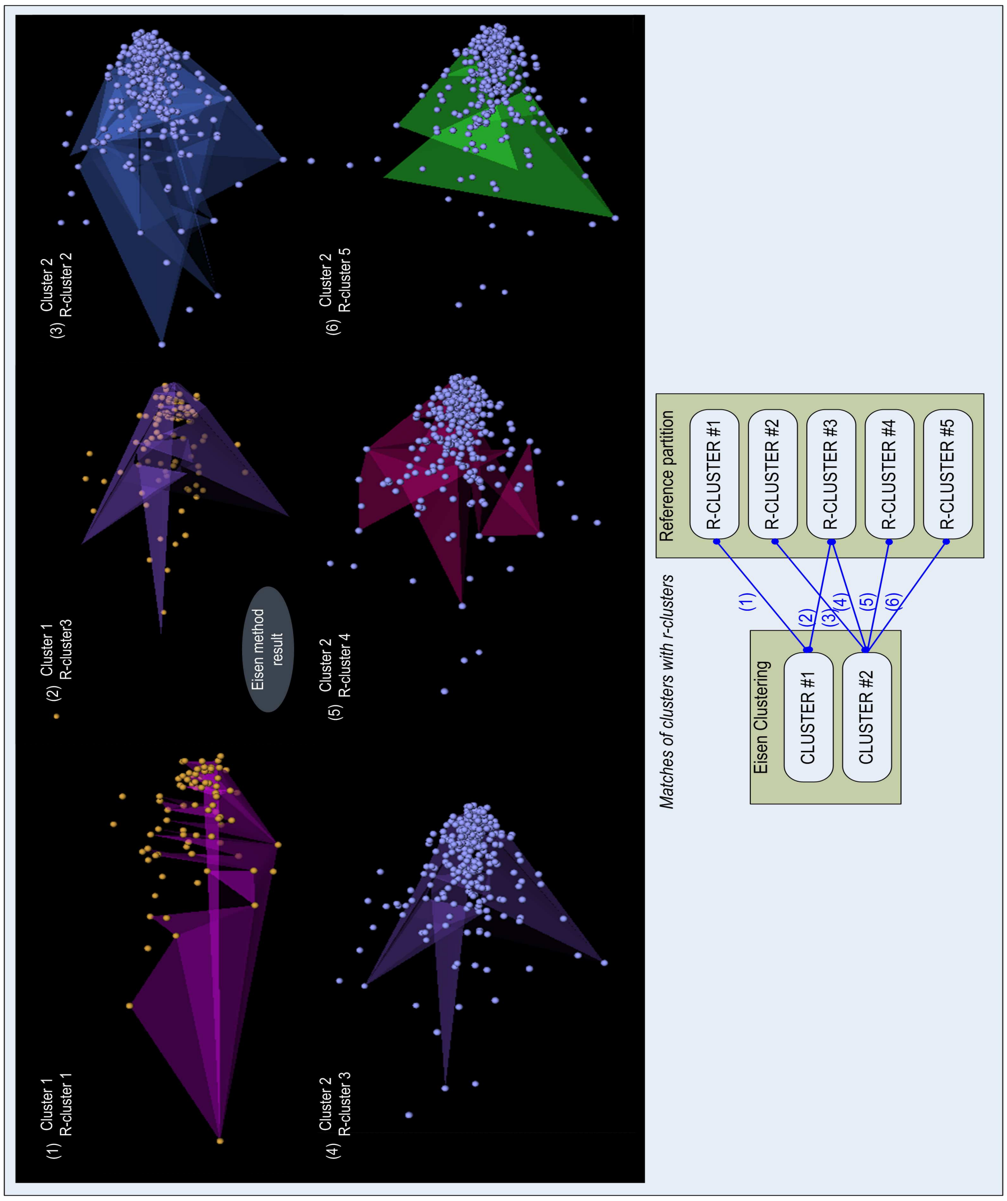


Figure 8.14: Shapes of the *cellcycle* reference partition and the clusters that better match each r-cluster, *Eisen* method.

sets, $TSVQ$ has been the method whose dendrogram has the best homogeneity, and the dendrogram with better separation has been for EMHC (see Tables 8.5 and 8.10).

8.5.1 sorlie Dendrograms

The goal of this experiment is to provide visual results as a means of comparison with the results given by the validity measures in the previous tables in Subsection 8.3.3. This way, also to show that some attributes (as for example, the number of clusters) in the results given by the validity measures in those comparative tables do not math their visualizations. This will prove that deciding on a clustering result, we need to apply more of one approach of validation, so that to decide on the global result from all approaches, which will improve the process of cluster analysis.

Consequently, Figures 8.15 and 8.16 show the dendrograms (on *sorlie* microarray) given by $TSVQ$ and EMHC respectively from Table 8.7. Figure 8.17 shows the data distribution of the *sorlie* data set in a 3D space. In Figures 8.15 and 8.16, clusters that visually fit data are represented through rectangles (in yellow color) on the microarray. Some areas in Figure 8.16 have not been highlighted because they are formed by very small or unitary clusters. The chosen clustering has been obtained not only by visual exploration of different views of the tool, but also considering the values given by the validity measures as a starting point. Note that in general, the number of clusters of the level visually chosen in these figures does not match the one analyzed in Table 8.8.

According to the gene-point distribution of the *sorlie* data set in Figure 8.17, *sorlie* is compact on the center of the data set and spread on the border. Therefore, since EMHC optimizes homogeneity and separation, it finds large clusters on the compact region of *sorlie* and small or unitary clusters on its spread region to reach its goal. This is shown in Figure 8.16, where there are large and small clusters unlike $TSVQ$ in Figure 8.15 that is focused only on homogeneity. Note that there are several small clusters on both, the left and the right side of the microarray in Figure 8.16, which are joined to the dendrogram at the end part of its construction (on top of the dendrogram). This means that EMHC is able to identify separations (outliers or noise) in the data set, fitting the data distribution given in Figure 8.17, which has not been possible for $TSVQ$.

In addition, two more figures showing the other dendrograms given by EMHC (on *sorlie*) from the two rows in Table 8.9 are respectively displayed in Figures 8.18

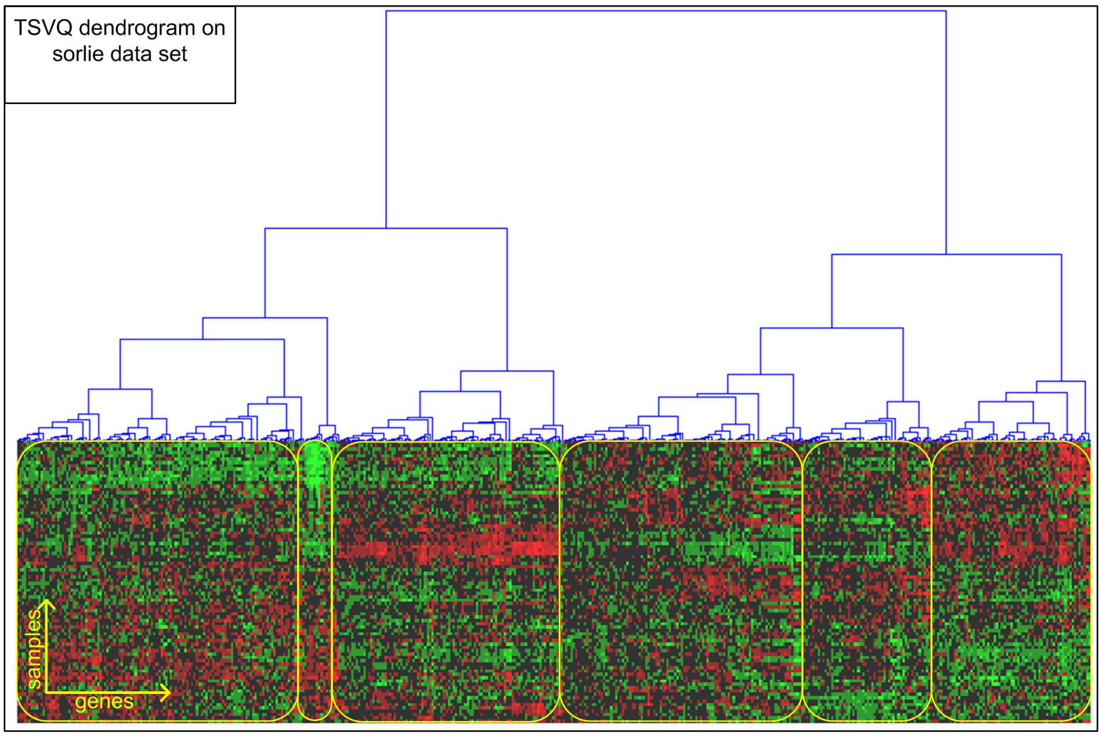


Figure 8.15: Visualization of dendrogram and microarray for TSVQ on *sorlie*. Clusters of the level selected in the dendrogram are highlighted.

and 8.19. As explained, small and unitary clusters in Figure 8.16 are the result of the optimization process of g_d (evaluates homogeneity and separation) to achieve its goal. Note that in Figure 8.18, most of these clusters have been removed due to the influence of the recombination of the solutions of the other methods (external solutions) taken as a starting point by EMHC. Thus, clusters (or the dendrogram) represented in Figure 8.18 are the result of g_d plus the inheritance and recombination of clusters from the other methods under the EMHC evolutionary strength.

The use of fitness function g_{rd} (evaluates homogeneity, separation and number of genes in the clusters) by EMHC on *sorlie* yields the result in Figure 8.19, where small clusters are fully removed from the evolutionary process. This confirms the homogeneity results reached in the second row in Table 8.9. Note that if unitary clusters (or small clusters) do not concern us, the dendrograms in Figures 8.18 and 8.19 represent then a

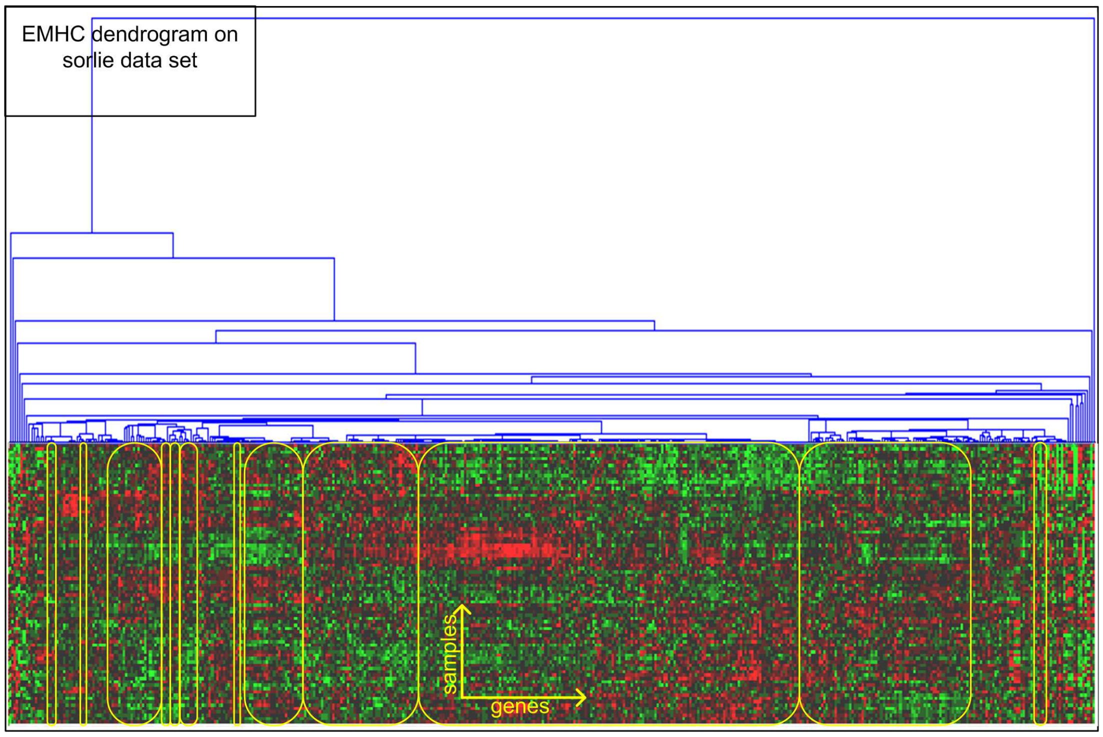


Figure 8.16: Visualization of the dendrogram and the microarray for EMHC on *sorlie*. Clusters of the level selected in the dendrogram are highlighted.

improvement of the dendrogram given in Figure 8.16. Moreover, according to all shown microarray views, the cluster distribution that better visually match the microarray of *sorlie* is that of EMHC in Figure 8.19. The above also shows that g_{rd} may be more useful than g_d performing on the *sorlie* data set.

8.5.2 lung Dendrograms

The goal of this experiment is the same as made for the *sorlie* data set in the above subsection but in this case, the experiment has been made on *lung* from the results given in the comparative tables of Subsection 8.3.4. The conditions of this experiment are the same as for *sorlie*, too.

Then, figures 8.20 and 8.21 show the dendrograms (on *lung* microarray) given by

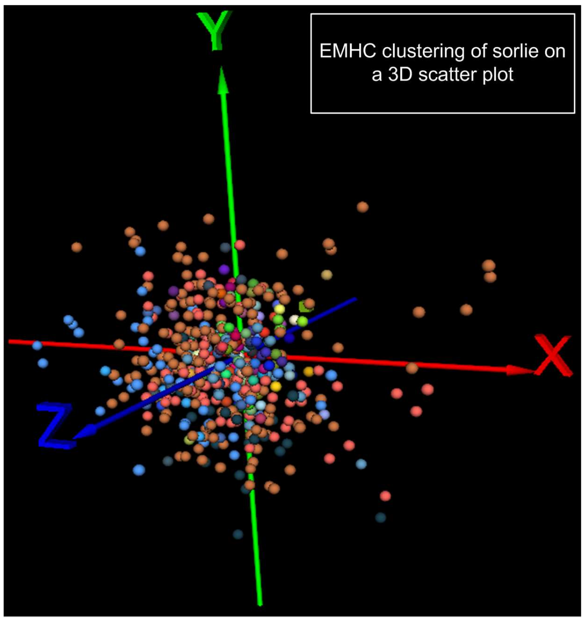


Figure 8.17: 3D scatter plot of the gene-points of *sorlie*.

TSVQ and EMHC respectively in Table 8.10. Figure 8.22 shows the data distribution of the *lung* data set in a 3D space. Note that it is more difficult to visually extract clusters from these *lung* microarrays than the microarrays previously given for *sorlie*. The answer is given by the gene distribution of *lung* shown on the scatter plot in Figure 8.22, gene-points shown in this figure show that *lung* is a very compact data set. A consequence of the compactness of *lung* is that, the homogeneity concept could fail since it is hard to separate (extract) clusters.

Even so, the distributions of clusters given in Figures 8.20 and 8.21 (clusters as yellow rectangles) for *TSVQ* and EMHC respectively are that better approximate the microarray (heap map) given by the *lung* data set in these figures. As shown for *sorlie*, EMHC has over again detected small clusters from *lung* that are separated from its compact region unlike *TSVQ*. This data set also shows that we should not focus only on either visualizations or statistical measures of the results, and that the best result can be reached by combining (optimizing) both items (visualizations and measures).

Summarizing about the performance of the EMHC method, we can say that it performs better than the others, on cluster separation measures and on measures that

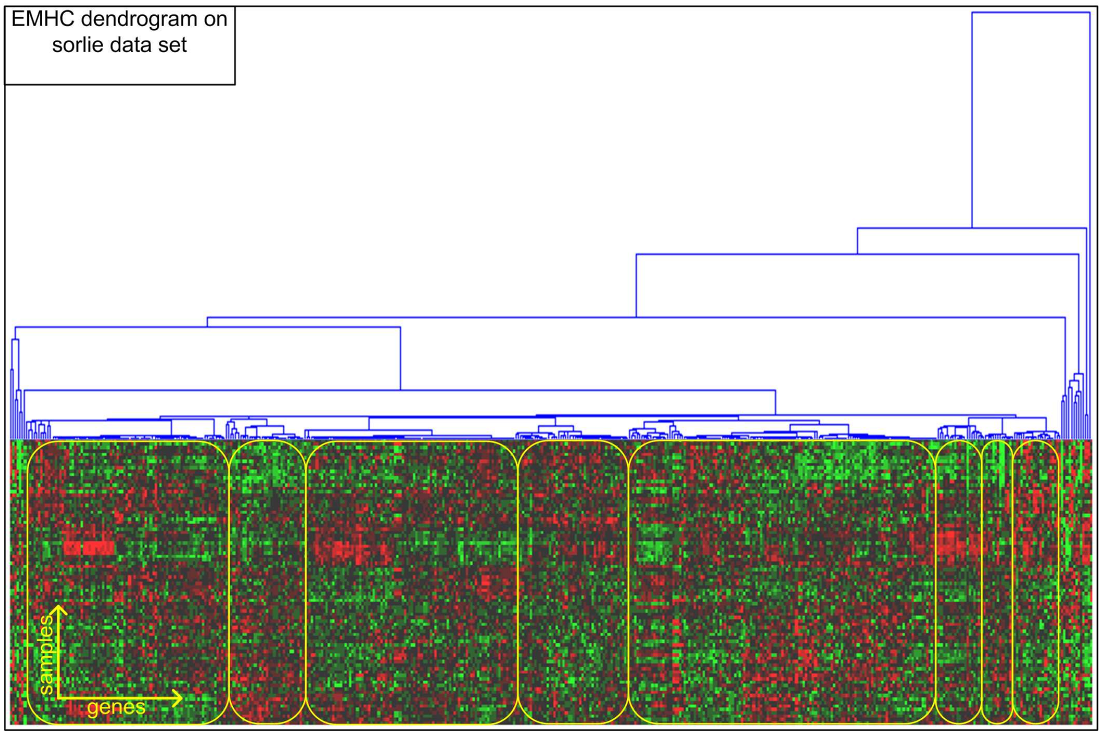


Figure 8.18: Visualization of dendrogram and microarray for EMHC (using g_d, see Definition 6.10) on *sorlie* from the solutions given by other methods. Clusters of the level selected in the dendrogram are highlighted.

combine homogeneity and separation (such as, silhouette width), but it does not have the same performance on homogeneity. It should be taken into account that most of the hierarchical clustering methods focus on cluster internal quality (homogeneity) and not on cluster external quality (adding separation), that is, global quality. For this reason, the other methods work better on homogeneity. However, EMHC looks for two or three indicators (fitness functions g_d and g_{rd}, see Definitions 6.10 and 6.14 in Chapter 6), homogeneity, separation and number of genes in each cluster, so that we can check two or three objectives on the clusters at a time. On the other hand, the previous visualizations of 3D-VC have also shown that EMHC performs well, even it can improve its solutions from the ones given by other methods.

Note that according to the parameter settings of EMHC and the kind of used data

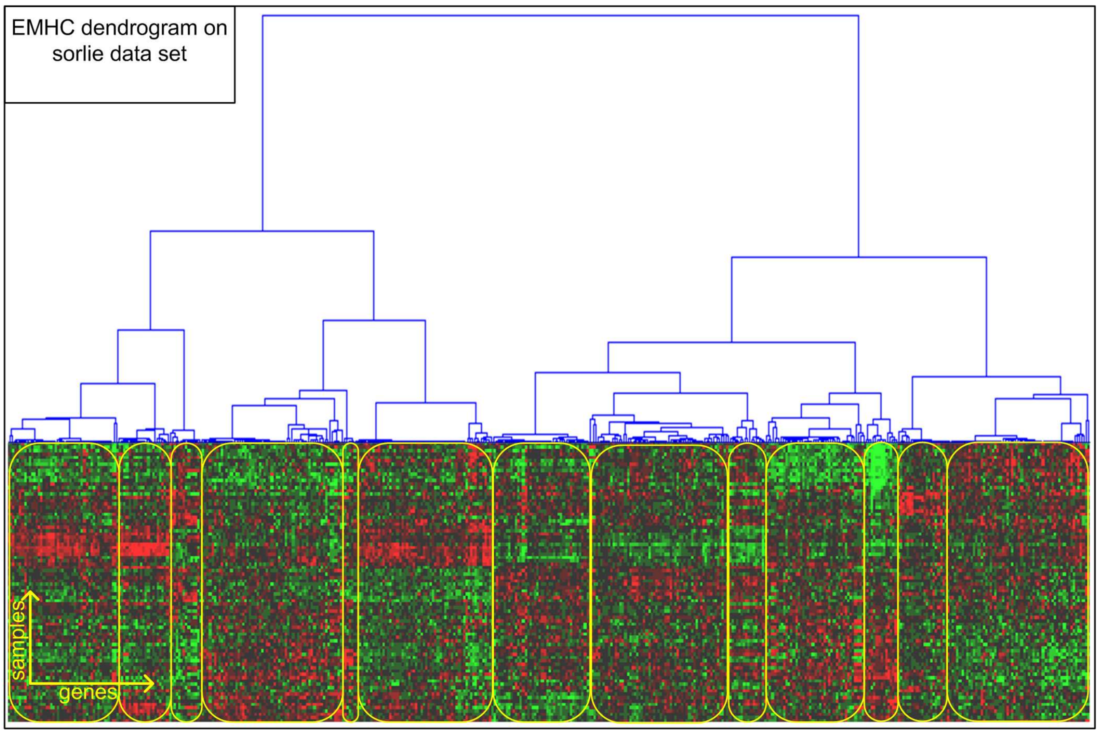


Figure 8.19: Visualization of dendrogram and microarray for EMHC (using g_{rd}, see Definition 6.14) on *sorlie* from the solutions given by other methods (external solutions). Clusters of the level selected in the dendrogram are highlighted.

set, EMHC can behave similar to the method that best has performed (after EMHC) on the analyzed data set. EMHC has performed similar to *Eisen* for *cellcycle* (see Figure 8.5), and *Agnes* for *sorlie* and *lung* (see Figures 8.6 and 8.7). The above can be the result of a possible adaptation of EMHC to the environment as a consequence of its evolutionary process.

To conclude, keep in mind that our goal has been to show that EMHC is an alternative with respect to existing methods able to find better solutions than them, not necessarily meaning that EMHC is the best in all aspects, a thing which would be unviable.

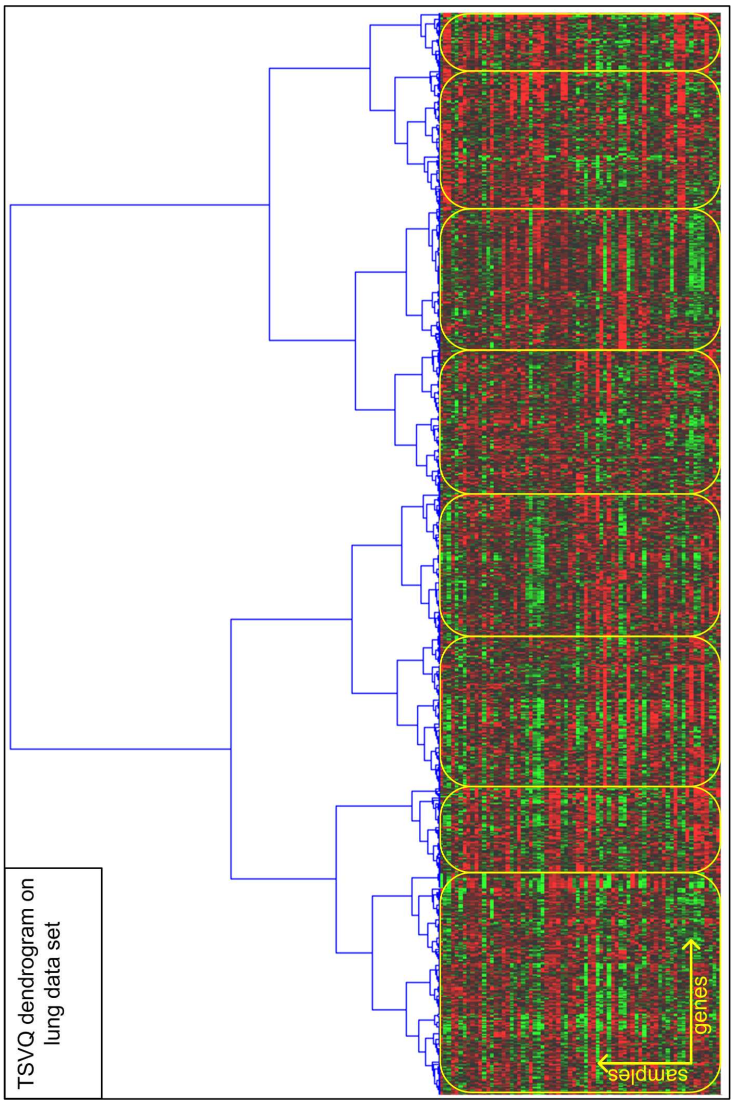


Figure 8.20: Visualization of dendrogram and microarray for *TSVQ* on *lung*. Clusters of the level selected in the dendrogram are highlighted.

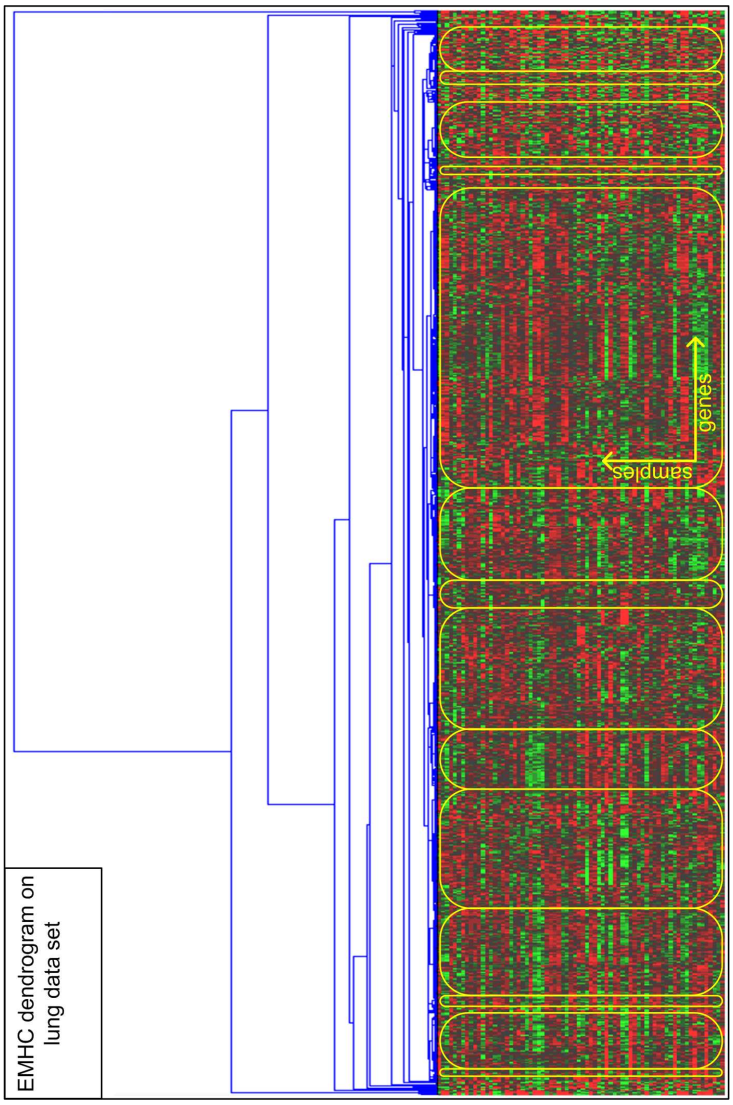


Figure 8.21: Visualization of dendrogram and microarray for EMHC on *lung*. Clusters of the level selected in the dendrogram are highlighted.

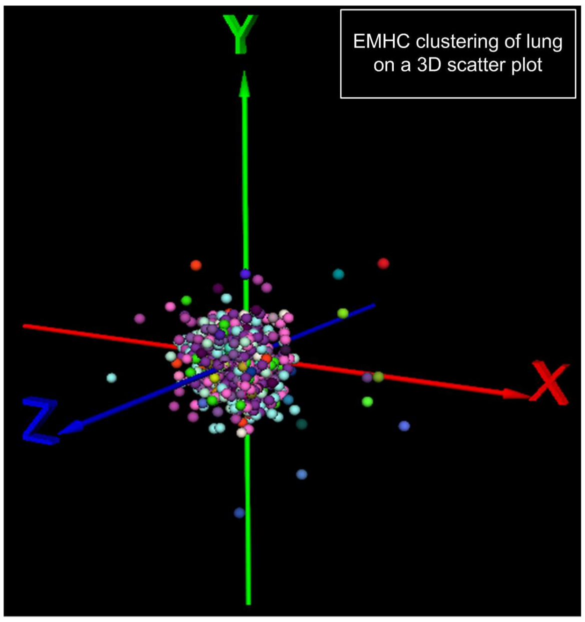


Figure 8.22: 3D scatter plot of the gene-points of *lung*.

Chapter 9

Conclusions

This research work has presented an evolutionary visual framework as global research which joins an evolutionary model of hierarchical clustering with an approach of visual analytics to increase the benefits obtained from the task of cluster analysis from DNA microarray data. As explained, our first goal has then been to define the evolutionary framework for cluster analysis from gene expression data. To reach that goal, several sub-goals have been outlined as stated in Chapter 1 and achieved in the following chapters of this document. According to this, the first chapter has evidenced the existing problems in cluster analysis and the need of new approaches that contribute to progress the study field.

The second chapter has introduced the data domain of our research, that is, DNA microarray data. Therefore, we have studied microarray technology and its main sources of biological knowledge available for their analysis. In that context, we have stated the main challenges in the analysis of DNA microarray data and their importance for human health. It should be noted on this point, that diagnosis from DNA microarray data is demanding new techniques and models able to support intelligent systems that can be used by specialists in clinical decision making (TBM, *Translational Biomedicine*).

Starting from the above, Chapter 3 has explained different techniques of unsupervised cluster analysis and Chapter 4 has studied evolutionary algorithms as a more recent technique used in cluster analysis of DNA microarray data. Chapter 3 has then presented the process of cluster analysis, a review of cluster validation techniques and the existing clustering methods with their performance on gene expression data. The need of visual validation of clustering results led us to also review the current visualization techniques related to DNA microarrays, and build a comparative table of visual components of the existing microarray visualization tools.

The analysis of DNA microarray data often involves complex processes, where several factors as high dimensionality of features, low number of samples or interrelated clusters among others, render this a hard task. In order to achieve better adaptation in such environment and to face those problems, in Chapter 4 we have introduced the theory of evolutionary algorithms and presented a review of their application to cluster analysis of DNA microarrays.

At this point, the first contribution according to our research hypothesis given in Chapter 1 has been to prove that *the problem of finding the best dendrogram from a data set is an NP-complete problem* (PFBD). This contribution has been given in Chapter 5 where the whole search space has also been characterized. Regardless of the theoretical and practical applications of this problem, classifying it to class NP-complete guarantees us that a good solution found to it is also good for any other problem in this class and conversely, as explained in Chapter 5.

Such previous results have motivated and justified the second part of our hypothesis, for which an evolutionary model of hierarchical clustering has been presented in Chapter 6. This model has developed a genetic algorithm along with several evolutionary heuristics, focused to deal with the previous problem PFBD. From the application of our model to cluster analysis of DNA microarray data emerges the need of using visualization techniques to validate its clustering results and compare them with other methods. In this context, Chapter 7 has introduced the third contribution of this research, the visual framework of cluster analysis from DNA microarrays. The practical result of this framework has been the development of a visualization tool called 3D-VisualCluster, which interacts with the previous evolutionary clustering model given in Chapter 6.

Finally, Chapter 8 has outlined all results achieved by our evolutionary framework as a fusion of the evolutionary model with visualizations of tool 3D-VisualCluster, from three public sets of DNA microarray data. As overall results, the fusion of both approaches to create the evolutionary framework has proved to be of great help in the understanding of the data and has provided knowledge about the used clustering methods. Moreover, we have provided new visualization components which can also be used to validate the existing ones, as shown in Table 3.1 of Chapter 3. From this evolutionary framework, it has also been proven that our evolutionary method of clustering performs well and that it can find better solutions than the other methods, shown not only through cluster validity measures but also, with visualizations of the results. This way, the visual part of our framework is able to display clustering results and validate both, cluster validity measures and more importantly, a visualization component with another one. Consequently, results have shown that our evolutionary framework is a powerful tool of cluster analysis from DNA microarray data, not only to our evolutionary model, but also for any other hierarchical clustering method that would want to add to the process. This way, note that the goals proposed in Chapter 1 have been met from Chapter 5 to Chapter 8.

9.1 Conclusion on EMHC

As a part of this research, we have proposed an evolutionary model aimed at hierarchical clustering building on DNA microarray data. By prefixing the parameters of this model, a specific method EMHC has been obtained, which provides new genetic operators able to evaluate and transform dendrograms according to the cluster definition. Several strategies (and constraints) have been introduced to reduce the complexity of the search space, that is, reduction of the level number of a dendrogram based on its non-valuable information (noise), reduction of the runtime of the fitness functions by introducing three fundamental lemmas, and the partition of the search space in neighborhoods to state a difference between local and global optimum. We have specifically introduced three genetic operators (two mutation operators and one crossover operator), performing an agglomerative and divisive strategy to build the child dendrograms from the mate process. In order to carry out an in-depth search to improve the solutions given by EMHC, several evolutionary strategies of local search have been defined. Formal details of all these issues have been given in Chapter 6.

The theoretical contributions introduced by our model (and the ones that also led to this model, Chapter 5) have allowed EMHC to find better solutions than other methods, as shown in the results given by the experimental evaluation on three public data sets containing DNA microarray data. Several reference clustering methods (Agnes, Diana, Eisen, HybridHclust and TSVQ) have been compared to the EMHC method based on internal and external cluster validity measures. As a general result, EMHC has performed well according to homogeneity and separation as well as according to a reference partition (JC measure, Jaccard Coefficient) of the chosen gene expression data set. Moreover, the experimental evaluation of the intrinsic key factors of the EMHC method (fitness function, mutation and crossover operators) has shown low sensitivity of EMHC with respect to some user parameters as crossover and mutation probabilities or the number of clusters, which is always a desirable feature of any software system. Another important result of our experiments has been that a genetic algorithm is not enough to deal with the problem of finding an optimum dendrogram in the search space. Hence, we have introduced several constraints and heuristics to make this problem tractable. Finally, EMHC can be installed in the R language ([16]) as a software-package, and is publicly available (under licence of R-Project, CRAN repository) along with the user manual at `http://cran.r-project.org/web/packages/clustergas`.

To conclude this section, note that our evolutionary model has also been conceived as an alternative to the existing methods of hierarchical clustering, which face difficulties such as:

- Most of hierarchical clustering methods assume that the proximity matrix does not contain ties (repeated values), otherwise the way in which these methods perform could turn out ambiguous [8, 9];

- Many of these methods follow a greedy algorithm strategy to build up dendrograms and do not offer the possibility of correcting errors found in the process [137];

- The biological data are very different from other kinds of data and the methods applied on them are of general purpose, a fact which leads to the need of proposing new methods able to capture the complex processes taking place at the cellular level [137, 210].

9.2 Conclusion on 3D-VisualCluster

The main goal of this research has been to provide an evolutionary framework for the analysis of gene expression data, where an evolutionary model (EMHC) has been joined to a visualization tool (3D-VisualCluster) to increase the performance of the knowledge discovery task on the data. To do this, we have first developed an approach of DNA microarray data analysis focused on the theory of metric spaces, which has allowed us to develop an algorithm to find cluster boundary gene-points. From the boundary gene-points, we have also defined another algorithm to approximate (reconstruct) the surface of a cluster and represent reference partitions of a data set in a 3D space.

The new visualizations of our visual model have been combined with other existing DNA microarray visualizations, such as heat maps, dendrograms and parallel coordinates, that the framework provides through linked views. This way, the visual analytics process can be achieved. Furthermore, we have defined a methodology to be followed by the user in both, visual cluster analysis and result validation, which is based on a set of tasks related to the visualizations given by the tool. On the other hand, since the 3D-VisualCluster tool has been linked to the R language [16], the results of clustering methods implemented in R can be visually analyzed by the tool. Moreover, as R is a programming language used in Bioinformatics, most clustering methods are implemented as software-packages in R.

In conclusion, we have decided to combine existing visualizations with the new ideas in our approach, so as to capitalize on the added value gained from interaction between these approaches and thus maximize the benefits to the user. A first prototype and additional material of tool 3D-VisualCluster have been developed and published at `http://www.analiticavisual.com/jcastellanos/3DVisualCluster/3D-VisualCluster`.

9.3 Future Work

The development of the evolutionary framework presented in this research has opened new expectations regarding the following step to continue this work. The future lines of research are, on one hand, aimed to study the behavior of other evolutionary optimization strategies on the problem addressed here, PFBD[1]. This way, the results reached by each used evolutionary strategy could be compared. In this context and since we have given a representation in form of a graph of the dendrogram search space on a data set in Chapter 5, and the algorithms of *ant colony system* ([79–81]) perform well on graph structures, it would be interesting to explore such an alternative in the proposed evolutionary framework. However, the application of algorithms of ant colony system to our problem is not trivial matter and so, this would start a new line of research.

On the other hand, even though our evolutionary framework introduces a new approach of visual cluster analysis from DNA microarray data, it does not yet include some partial knowledge from experts of the problem domain. Since the selection of a specific clustering algorithm and the cluster validity measures in cluster analysis is usually obtained by combination of some evaluation criterium and the user experience [137], we can then couple such information (expert knowledge) to the evolutionary framework to reduce the user effort and improve the understanding of both, the data and applied algorithms. However, this could turn the cluster analysis task into a supervised process.

[1] *The problem of finding the best dendrogram on a data set.*

Chapter 9

Conclusiones

En este trabajo de investigación se ha presentado un *framework* visual evolutivo como investigación global. Este *framework*, fusiona un modelo evolutivo de *clustering* jerárquico con un enfoque de analítica visual que permite incrementar el rendimiento de la tarea del análisis de *cluster* sobre datos de *DNA-microarrays*. Como se ha explicado, nuestro primer objetivo ha sido, entonces, definir el *framework* evolutivo para el análisis de *cluster* a partir de datos de expresión génica. Para llevar a cabo esta tarea, se han definido varios sub-objetivos tal y como se han enumerado en el Capítulo 1. La consecución de cada uno de ellos se ha descrito en los diferentes capítulos de esta memoria de tesis. Así pues, en el primer capítulo, se enunciaron los problemas existentes en el análisis de *cluster* y la necesidad de nuevos enfoques que contribuyan a progresar en el campo de estudio.

Por otra parte, en el segundo capítulo, se introdujo el dominio de aplicación sobre el cual hemos investigado, el análisis de datos procedentes de *DNA-microarrays*. Este capítulo se ha centrado en describir la tecnología de *microarrays* y en este contexto, se establecieron los principales retos en el análisis de datos de *DNA-microarrays* y su importancia para la salud humana. Cabe destacar sobre este aspecto, que la diagnosis a partir de datos de *DNA-microarrays* está demandando nuevas técnicas y modelos, capaces de integrar sistemas inteligentes que puedan ser empleados por los especialistas en la toma de decisión clínica (TBM, *Translational Biomedicine*).

En consecuencia con lo anterior, en el Capítulo 3, se analizaron las técnicas del análisis de *cluster* no supervisado, y en el Capítulo 4, se estudiaron los algoritmos evolutivos como una de las técnicas más recientes en el análisis de *cluster* sobre datos de *DNA-microarrays*. En primer lugar, en el Capítulo 3 se presentaron: el proceso del análisis de *cluster*, una revisión de las técnicas de validación de *cluster* y los métodos de *clustering* existentes, así como, su rendimiento sobre datos de expresión génica. La necesidad de validar visualmente los resultados de un *clustering*, nos condujo también, a hacer una revisión de técnicas de visualización relacionadas a *DNA-microarrays* y a construir una tabla comparativa de componentes visuales, actualmente empleados por las herramientas de visualización de *microarrays*.

El análisis de datos provenientes de *DNA-microarrays*, con frecuencia, involucra procesos complejos, donde factores como, la alta dimensionalidad de las características, el bajo número de muestras o la presencia de *clusters* interrelacionados hacen que esta tarea sea verdaderamente difícil. Con el objetivo de lograr una mejor adaptación a este dominio de aplicación y enfrentar tales problemas, hemos introducido en el Capítulo 4, la teoría de los algoritmos genéticos y hemos presentado también, una revisión de su aplicación al análisis de *cluster* sobre datos de *DNA-microarrays*.

A partir de aquí, la primera contribución relacionada con nuestra hipótesis de trabajo primario, se basó en demostrar que *el problema de la búsqueda del mejor dendograma sobre un conjunto de datos, es un problema NP-completo* (PFBD). la demostración formal de esta hipótesis aparece en el Capítulo 5, aportándose también una caracterización formal del espacio de búsqueda asociado a este problema. Independientemente de las aplicaciones, tanto prácticas como teóricas de este problema, clasificarlo en la clase NP-completo, nos garantiza, que una buena solución encontrada a este problema, sería una buena solución también, para cualquier problema de esta clase y viceversa, tal como se ha explicado en el Capítulo 5.

Por otra parte, los resultados anteriores, motivaron y justificaron la segunda parte de nuestra hipótesis, para lo cual, en el Capítulo 6 se desarrolló un modelo evolutivo de *clustering* jerárquico para resolver el problema PFBD y alcanzar soluciones subóptimas. Bajo este modelo se desarrolló un algoritmo genético junto con varias heurísticas evolutivas, todo ello, enfocado a encarar el problema PFBD. De la aplicación de nuestro modelo al análisis de *cluster* de datos provenientes de *DNA-microarrays*, surgió la necesidad de emplear técnicas de visualización para validar los resultados de *clustering* y compararlos con los de otros métodos. En este sentido, en el Capítulo 7, se introdujo la tercera contribución de este trabajo de investigación, el modelo visual para el análisis de *cluster* sobre datos de *DNA-microarrays*. El resultado práctico de tal modelo fue el desarrollo de una herramienta de visualización de datos de *microarrays*, denominada 3D-VisualCluster, la cual es capaz de interactuar con el modelo evolutivo de *clustering* definido en el Capítulo 6.

Finalmente, en el Capítulo 8 se describieron todos los resultados alcanzados por nuestro *framework* evolutivo, como una fusión del modelo evolutivo con las visualizaciones de la herramienta 3D-VisualCluster a partir de tres conjuntos de datos de *DNA-microarrays*. Como resultados globales tenemos que, la fusión de ambos enfoques para crear el *framework* evolutivo, resultó de gran ayuda en la comprensión de los datos, proporcionando también, conocimiento acerca de los métodos de *clustering* empleados. Además, se han introducido nuevos componentes de visualización, los cuales pueden utilizarse para validar otras visualizaciones existentes, tal como las mostradas en la Tabla 3.1 del Capítulo 3. A partir de este *framework* evolutivo, se ha demostrado también, que nuestro método evolutivo de *clustering* presenta un buen desempeño y que éste es capaz de encontrar mejores soluciones que los otros métodos. Esto no sólo fue mostrado a través de las medidas de validación de *clusters*, sino también, con visualizaciones de los resultados. De esta manera, la parte visual de nuestro *framework*, es capaz de visualizar resultados de los *clusterings* calculados, evaluar visualmente las medidas de validez de *cluster* y lo que es más importante, validar un componente de visualización con otro. En consecuencia, los resultados expuestos permiten demostrar que el *framework* evolutivo propuesto, es una poderosa herramienta para el análisis de *cluster* de datos provenientes de *DNA-microarrays*, y que no sólo lo es para nuestro modelo evolutivo, sino también para cualquier método de *clustering* jerárquico que desee incluir al proceso. Nótese que de esta forma se han alcanzado los objetivos propuestos en el Capítulo 1, tal y como se ha descrito desde el Capítulo 5 hasta el Capítulo 8 de esta memoria.

9.1 Conclusión sobre EMHC

Como ya se ha mencionado, en este trabajo de investigación se ha propuesto un modelo evolutivo, orientado a la construcción de *clusterings* jerárquicos a partir de datos de *DNA-microarrays*. Prefijando los parámetros de este modelo, se obtuvo el método específico EMHC, el cual proporciona nuevos operadores que evalúan y transforman dendogramas, basándose en la propia definición de *clusterings*. Asimismo, se han introducido estrategias and restricciones para reducir la complejidad del espacio de búsqueda. En concreto, estas estrategias se han centrado en: i) la reducción del número de niveles de los dendogramas mediante la poda de aquellos niveles con información no valiosa (ruido), ii) la reducción del tiempo de ejecución de las funciones de aptitud a partir de tres lemas fundamentales, y iii) la partición del espacio de búsqueda en vecindades para establecer las diferencias entre óptimo local y óptimo global. Específicamente, hemos introducido tres operadores genéticos (dos operadores de mutación y un operador de cruce), los cuales desarrollan estrategias aglomerativas y divisivas en la construcción de los dendogramas hijos durante la etapa de reproducción. Con el objetivo de realizar una búsqueda en profundidad para mejorar las soluciones encontradas por el método EMHC, se han desarrollado varias estrategias evolutivas de búsqueda local. Los detalles sobre cada una de las características anteriores, todas ellas relativas al modelo evolutivo, se han detallado en el Capítulo 6.

Las características soportadas por los resultados teóricos expuestos en el Capítulo 5 e incluidas en el modelo propuesto, han permitido a EMHC, encontrar mejores soluciones que el resto de los métodos. Esto se ha demostrado, a través de los resultados alcanzados en la evaluación experimental realizada sobre tres conjuntos de datos de *DNA-microarrays*. Varios métodos de *clustering* de referencia (Agnes, Diana, Eisen, HybridHclust and TSVQ) se compararon con EMHC, basándonos en la evaluación de las medidas de validez de *cluster*, tanto internas como externas. Como resultado general, tenemos que el funcionamiento de EMHC es bueno respecto a homogeneidad, separación y a una partición de referencia (medida JC, *Jaccard Coefficient*) del conjunto de datos de expresión génica seleccionado. Aún más, la evaluación experimental de los factores claves e intrínsecos de EMHC (función de aptitud, operadores de mutación y cruce) han demostrado poca sensibilidad respecto a algunos parámetros definidos por el usuario, como son las probabilidades de cruce y mutación, el número de *clusters* (lo cual es siempre una característica deseable para cualquier software). Otro resultado, no menos importante de nuestros experimentos, es que un algoritmo genético por sí solo, no es suficiente para abordar el problema de encontrar un dendograma óptimo en el espacio de soluciones de todos los posibles dendogramas. Por esta razón, se han definido varias restricciones y heurísticas para convertir este problema intratable en un problema tratable. Finalmente, EMHC puede ser instalado en R ([16]) como un paquete-software, y está públicamente disponible (bajo licencia de *R-Project*, repositorio CRAN) junto con el manual del usuario en `http://cran.r-project.org/web/packages/clustergas`.

Para concluir este apartado, se quiere hacer notar que el modelo evolutivo propuesto se ha desarrollado también, como una alternativa respecto a los métodos existentes de *clustering* jerárquico, los cuales presentan dificultades como:

- La mayoría de los métodos de *clustering* jerárquicos, asumen que la matriz de proximidad no contiene valores repetidos, en otro caso, el funcionamiento de estos métodos pudiera ser ambiguo [8, 9];
- Muchos de estos métodos, siguen una estrategia voraz para construir los dendogramas, no dando la posibilidad de corregir posibles errores en el proceso [137];
- Los datos biológicos suelen ser muy diferentes del resto de datos, por lo que, los métodos aplicados sobre ellos son de propósito general, lo cual nos conduce a proponer nuevos métodos con la capacidad de capturar los complejos procesos ocurridos a nivel celular [137, 210].

9.2 Conclusión sobre 3D-VisualCluster

El principal objetivo de esta investigación a nivel práctico ha sido brindar un *framework* evolutivo, orientado al análisis de datos de expresión génica, donde un modelo evolutivo (EMHC) se une a una herramienta de visualización (3D-VisualCluster), con el objetivo de incrementar el rendimiento de la tarea de extracción de conocimiento a partir del análisis de los datos. Con este fin, se ha desarrollado un enfoque de análisis visual de datos de *DNA-microarrays*, basado en la teoría de los espacios métricos, lo cual nos ha permitido construir un algoritmo para encontrar los puntos (genes) frontera de un *cluster*. A partir de los puntos frontera, se pudo construir otro algoritmo para aproximar (reconstruir) la superficie de un *cluster* y representar particiones de referencia de un conjunto de datos, todo en el espacio tridimensional.

Las nuevas visualizaciones propuestas se combinan con otras visualizaciones de *DNA-micoarrays* existentes, tales como, mapa de colores (*heat map*), dendogramas y coordenadas paralelas, que nuestro *framework* ofrece a través de vistas enlazadas dentro de la aplicación desarrollada. De esta forma, se dispone de una herramienta-software para realizar el proceso de analítica visual. Adicionalmente, se definió una metodología a seguir por el usuario en el análisis visual de *cluster* y la validación de los resultados, a partir de un conjunto de tareas relacionadas a las visualizaciones ofrecidas por la herramienta. Por otra parte, y dado que la herramienta 3D-VisualCluster está enlazada al lenguaje R [16], los resultados de cualquier método de clustering implementado en R, puede analizarse visualmente con esta herramienta. Además, como R es un lenguaje de programación ampliamente utilizado en el campo de la Bioinformática, la mayoría de los métodos de *clustering* están implementados en R, a través de los correspondientes paquetes de software desarrollados por la comunidad, y *a priori*, son utilizados por parte de la herramienta desarrollada.

En conclusión, se ha decidido combinar la visualizaciones existentes con las contribuciones de este trabajo de investigación, con el objetivo de resaltar el valor añadido proveniente de la interacción entre enfoques distintos y, por tanto, maximizar los beneficios de cara al usuario final. Un primer prototipo de la herramienta 3D-VisualCluster, así como, información adicional se puede encontrar en `http://www.analiticavisual.com/jcastellanos/3DVisualCluster/3D-VisualCluster`.

9.3 Trabajo Futuro

El desarrollo del *framework* evolutivo presentado en esta investigación, ha abierto nuevas fases y expectativas para la continuación de este trabajo. Las líneas futuras de investigación están orientadas, por una parte, al estudio del comportamiento de otras estrategias de optimización evolutiva sobre el problema investigado en este trabajo, PFBD[1]. De esta forma, podríamos comparar los resultados alcanzados por las diferentes estrategias evolutivas empleadas. En este contexto y del hecho de haber dado en el Capítulo 5, una representación del espacio de búsqueda de los dendogramas en forma de grafo, y de que los algoritmos de *sistemas de colonias de hormigas* ([79–81]) tienen un buen funcionamiento sobre estructuras de grafos; sería interesante explorar tal alternativa en el *framework* evolutivo propuesto. Sin embargo, la aplicación de los sistemas de colonias de hormigas a nuestro problema, no es una tarea trivial y por tanto, esto podría generar una nueva línea de investigación.

[1] *El problema de encontrar el mejor dendograma sobre un conjunto de datos.*

Por otra parte, aún cuando nuestro *framework* evolutivo, introduce un nuevo enfoque de análisis de cluster visual sobre datos de *DNA-microarray*, éste no incluye conocimiento parcial de expertos en el dominio del problema. En el ámbito del análisis de *cluster*, la elección de un algoritmo específico de *clustering* y las medidas de validez de los *clusters* obtenidos, se establecen normalmente mediante la combinación de algún criterio de evaluación y la experiencia del usuario [137]. Entonces, se puede plantear la posibilidad de incluir información (conocimiento experto) al *framework* evolutivo para reducir el esfuerzo del usuario, además de mejorar la comprensión de los datos y de los algoritmos aplicados. No obstante, hay que tener en cuenta que este enfoque, pudiera convertir el proceso del análisis de *cluster* no supervisado, en supervisado.

Appendix A

Scientific Works, Conferences and Publications of this Research

- Castellanos-Garzón, J. A. & Díaz, F. (2012), *An Evolutionary Computational Model Applied to Cluster Analysis of DNA Microarray Data*, Submitted to Expert Systems with Applications, Elsevier.

- Castellanos-Garzón, J. A.; García, C. A.; Novais, P. & Díaz, F. (2012), *A Visual Analytics Framework for Cluster Analysis of DNA Microarray Data*, Expert Systems with Applications, Elsevier. In Press - `http://dx.doi.org/10.1016/j.eswa.2012.08.038(0)`.

- Castellanos-Garzón, J. A. & Díaz, F. (2012), *clustergas: A hierarchical clustering method based on genetic algorithms*, R package version 1.0, CRAN R-Project. Department of Computer Science, University of Valladolid (Spain). [online] `http://cran.r-project.org/web/packages/clustergas`.

- García, C. A. & Castellanos-Garzón, J. A. (2011), *Analyzing Gene Expression Data on a 3D Scatter Plot*, Advances in Intelligent and Soft Computing, Springer-Verlag 87, 349-356.
  Presented in: International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO'11).

- Castellanos-Garzón, J. A.; García, C. A. & Miguel-Quintales, L. A. (2009), *An Evolutionary Hierarchical Clustering Method with a Visual Validation Tool*, Lecture Notes on Computer Science, Springer-Verlag 5517, 367-374.
  Presented in: 10th International Work-Conference on Artificial Neural Networks (IWANN2009).

- Castellanos-Garzón, J. A. & Miguel-Quintales, L. A. (2008), *Evolutionary Techniques for Hierarchical Clustering Applied to Microarray Data*, Advances in Soft Computing, Springer Berlin / Heidelberg 49, 118-127.
  Presented in: 2nd International Workshop on Practical Applications of Computational Biology & Bioinformatics (IWPACB2008).

- Castellanos-Garzón, J. A. & Miguel-Quintales, L. A. (2007), *Un Método de Clustering Jerárquico con Algoritmos Genéticos: un Caso Práctico*, I Workshop Español sobre Extracción y Validación de Conocimiento sobre Bases de Datos Biomédicas (EVaBio'07), XII Conferencia de la Asociación Española para Inteligencia Artificial (CAEPIA-TTIA), ISBN-13:978-84-611-8854-3.

- Castellanos-Garzón, J. A. & Miguel-Quintales, L. A. (2007), *Un Método de Clustering Jerárquico Basado en Algoritmos Genéticos*, IEEE Computational Intelligence Society (SC), II Simposio de Inteligencia Computacional (SICO'07), II Congreso Español de Informática (CEDI), Editorial Thomson ISBN: 978-84-9732-606-3, S15, 177-185.

- Castellanos-Garzón, J. A. (2006), *Algoritmos Genéticos para Clustering de Datos de Expresión Génica.* Masters Thesis, Department of Computer Science and Automatics, University of Salamanca (Spain).

Appendix B

Data Dimensionality Reduction

Dimensionality reduction algorithms ([93, 98, 245]) map a data set of n rows of a d-dimensional space on an m-dimensional space, where $m < d$. The aim is to project the data onto a low dimensional space so as to retain as much structure as possible.

Linear projections express m features as linear combinations of the original d features; that is, $\mathrm{y}_i = \mathfrak{M} \cdot \mathrm{x}_i$, where y_i is an m-place column vector, $\mathfrak{M}$ is a matrix of $m \times d$ and x_i is a d-place column vector, $i \in [1, n]$. Linear projection methods preserve the character of the data. One of the techniques most commonly used to linearly reduce the dimensionality of a space is *principal component analysis* (PCA) [77,142,219], which is used in our framework.

B.0.1 Principal Component Analysis

Given X denoting an $n \times d$ matrix of real-valued gene expression data [137], and x_{ij} the expression level of the i^{th} gene in the j^{th} condition. If $\mathfrak{C}$ is the condition covariance matrix of X, then the eigenvectors of $\mathfrak{C}$ define a linear projection that replaces the features in the raw data with uncorrelated conditions. These eigenvectors also provide a link between cluster analysis and *factor analysis* [8,35]. Since $\mathfrak{C}$ is a $d \times d$ positive definite matrix, its eigenvalues are real and can be computed via the equation $|\mathfrak{C} - \lambda \cdot \mathrm{I}| = 0$, being $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$.

A set of corresponding eigenvectors, $\{\mathrm{c}_1, \mathrm{c}_2, \ldots, \mathrm{c}_d\}$ is ordered accordingly, where every eigenvector satisfies the following condition $\mathfrak{C} \cdot \mathrm{c}_j = \lambda_j \cdot \mathrm{c}_j, j \in [1, d]$. Eigenvectors are nonzero vectors which form a d-dimensional orthonormal basis that, when a linear transformation is applied to them, may change in length but not direction.

The $m \times d$ matrix of transformation $\mathfrak{M}_m$ is built by transposition of the first m eigenvectors of the covariance matrix (either correlation matrix or least square matrix) $\mathfrak{C}$, namely $\{\mathrm{c}_1^t, \mathrm{c}_2^t, \ldots, \mathrm{c}_m^t\}$. Eigenvectors are also called principal components.

$\mathfrak{M}_m$ projects the gene space into an m-dimensional subspace whose axes are in the directions of largest eigenvalues of $\mathfrak{C}$:

$$\mathrm{y}_i = \mathfrak{M}_m \cdot \mathrm{x}_i, i \in [1, n]. \tag{B.0.1}$$

The genes projected can be written as:

$$\mathfrak{P}_m = \begin{bmatrix} \mathrm{y}_1^t \\ \mathrm{y}_2^t \\ \vdots \\ \mathrm{y}_n^t \end{bmatrix} = X \cdot \mathfrak{M}_m^t. \tag{B.0.2}$$

Note x_i as a row vector of X is the original gene (or object) and y_i is the corresponding projected gene. Equation (B.0.1) is called *eigenvector transformation* (or eigengene transformation, in our case).

Projection of data into singular value decomposition subspaces and visualization with scatter plots can reveal structures in the data that will be used in the classification process. In the case of gene expression analysis, one may want to cluster conditions in a diagnostic study or cluster genes in a systems biology study.

As an exploratory data analysis technique, PCA can be used to detect outliers, uncover data structures that account for a large percentage of the total variance and create new hypothetical constructs that may be employed to predict or classify observations into clusters.

Appendix C

Metric Spaces

Definition C.1. *Metric spaces.*
If E is an arbitrary set and $\rho : E \times E \to \mathbb{R}$ a mapping called the distance, then the pair $< E, \rho >$ (abbreviated to E) is a metric space if ρ is a metric defined on E. That is, for all points $x, y, z \in E$, the following are satisfied:

1. Positiveness: $\rho(x, y) > 0$ if $x \neq y$, and $\rho(x, x) = 0$.
2. Symmetry: $\rho(x, y) = \rho(y, x)$.
3. Triangle inequality: $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$.

Definition C.2. *Bounded metric spaces.*
Let E be a metric space consisting with the metric ρ. If there exists a positive number k such that $\rho(x, y) \leq k$ for all points $x, y \in E$, then we say that E is a *bounded* metric space.

Definition C.3. *Diameter of a subset of a metric space.*
Let A be a subset of a metric space E. Consider the set of non-negative real numbers $\{\rho(x, y) \mid x \in A, y \in A\}$. If this set is bounded, then it has supremum, denoted $\delta(A)$, which is called the diameter of A.

Definition C.4. *Distance between two subsets.*
Let A and B be two subsets of a metric space E. The set of real numbers $\{\rho(x, y) \mid$

$x \in A, y \in B\}$, is bounded below by zero. Its infimum, denoted by $\rho(A, B)$, is called the distance between A and B.

In the case that A and B can be considered as clusters, other definitions of distance between them can be given [8].

Definition C.5. *Subspace of a metric space.*
Let E be a metric space with the metric ρ, and let E' be a proper subset of E. Let ρ' be the restriction of ρ to $E' \times E'$, so that $\rho'(x, y) = \rho(x, y)$ for all x and y in E'. Then ρ' is called an *induced metric* and E' with the metric ρ' is called a *subspace of* E.

The gene expression matrix $(m \times n)$ of a DNA microarray can be viewed as a subspace of $\mathbb{R}^n$ (see Proposition D.1, Appendix D).

Definition C.6. *Balls.*
If a is a point in a metric space E with the metric ρ, then the set of all points $x \in E$ such that $\rho(a, x) < r$, where $r > 0$, is called a *ball* (or neighborhood) of centre a and radius r, and is denoted by $N(a, r)$.

Definition C.7. *Open sets.*
If A is a subset in a metric space E, the point $a \in A$ is said to be an *interior point* of A if it is the centre of a ball which consists only of points in A. The set of all interior points of A (Int$A$) is called the *interior* of A. If every point of A is interior, then A is called an *open set.*

Definition C.8. *Adherent points.*
If A is a subset of a metric space E, the point $a \in E$ is said to be an *adherent point* of A if every ball with centre a contains a point of A.

Definition C.9. *Closure.*
If A is a subset of a metric space E, the *closure* of A, denoted by $\overline{A}$, is the set of all adherent points of A. Clearly, $A \subset \overline{A}$. If $A = \overline{A}$, we say that A is *closed.*

Definition C.10. *Exterior points.*
A point is said to be an *exterior point* of A if it is an interior point of the complement A^c. The exterior of A (Ext A) is the set of all exterior points of A.

Definition C.11. *Frontier points.*
The set $\overline{A} \cap \overline{A^c}$, which is not necessarily empty, is called the *frontier* of A, denoted FrA. The *boundary* of a set A, denoted BdA, is the part of the frontier of A which belongs to A.

Appendix D

Theoretical Results of Chapter 7

Proposition D.1. *Gene subspace.*
Consider the metric space $\mathbb{R}^d$ and without loose of generality, the Euclidean distance ρ' defined on $\mathbb{R}^d$. Let $X_{n \times d}$ be a gene expression matrix of DNA microarray data, and let $\mathfrak{G}_n^d$ be a set consisting of all row points (genes) in $X_{n \times d}$. Then $\mathfrak{G}_n^d$ is a bounded subspace of $\mathbb{R}^d$ with the induced metric ρ with $\rho(g_x, g_y) = \rho'(g_x, g_y)$, for g_x and g_y genes in $\mathfrak{G}_n^d$.

Proof: This proposition can be easily verified from definitions C.1, C.2 and C.5 from Appendix C.

Proposition D.2. *Closed cluster.*
If $\mathfrak{G}_n^d$ is a gene metric space with a metric ρ, then every cluster C of $\mathfrak{G}_n^d$ is a closed set (*closed cluster*) in $\mathfrak{G}_n^d$.

Proof: From Definitions C.8 and C.9, it is necessary to prove that $C = \overline{C}$. We do this by *reductio ad absurdum*. Suppose that $x \in \mathfrak{G}_n^d$ is an exterior gene of C (see Definition C.10) that is also an adherent gene of C. Then $x \in \overline{C}$ and $C \neq \overline{C}$, and moreover, $\forall r \in \mathbb{R}, \exists a \in C, a \in N(x, r)$ (see Definition C.6). If, in particular, $r = \rho(x, C)/2$ (see Definition C.4), then $\nexists a \in C$ such that $a \in N(x, \rho(x, C)/2)$, and so $x \notin \overline{C}$, which is a contradiction to our previous assumption. Hence, $x \in \overline{C}$, $C = \overline{C}$ and thus C is closed. $\square$

Definition D.1. *Extreme gene.*
Let C be a cluster of a gene metric space $\mathfrak{G}_n^d$. A gene $g \in C$ such that $g = (x_1, x_2, \ldots, x_i, \ldots, x_d)$ is said to be an i^{th} *extreme gene* (or an extreme gene) of C, $i \in [1, d]$, if either $x_i \geq x'_i$ or $x_i \leq x'_i$ $\forall g' = (x'_1, x'_2, \ldots, x'_i, \ldots, x'_d) \in C \setminus \{g\}$. An i^{th} extreme gene $g \in C$ is denoted as $g^i_{\succ}$ when $x_i \geq x'_i$ or $g^i_{\prec}$ when $x_i < x'_i$ in the above mentioned condition. The set of all extreme genes of C is denoted by Exm$C$.

Proposition D.3. *Extreme genes and cluster boundary.*
If $g = (x_1, x_2, \ldots, x_i, \ldots, x_d)$ is an i^{th} extreme gene of a cluster C of $\mathfrak{G}_n^d$, then $g \in \mathrm{Bd}C$.

Proof: It is enough to prove that $g \in C$ is not an interior gene of C. This is true because in any ball $N(g, r)$ with $r > 0$, we can find a gene $g' = (x'_1, x'_2, \ldots, x'_i, \ldots, x'_d)$ with either $x'_i > x_i$ or $x'_i < x_i$ (according to the case) such that g' is an i^{th} extreme gene of $C \cup \{g'\}$ and $\rho(g', g) < r$, $i \in [1, d]$. This implies that $g' \in N(g, r)$ and $g' \notin C$, so g is not an interior gene in C and hence $g \in \mathrm{Bd}C$. □

Proposition D.4. *Cardinality of P_m.*
Since P_m is the set of mid-points, where $2d$ extreme points are achieved from an iteration of ClusterBoundary, the cardinality of P_m is $Card(P_m) = 2d(d-1)$.

Proof: The number of mid-points computed from extreme points, and that satisfy module (III) of the algorithm, can be expressed as $4\sum_{i=1}^{d-1}(d-i)$. Expanding this expression, the required result is reached. □

Proposition D.5. *Runtime of ClusterBoundary*
The temporal complexity of ClusterBoundary for computing the cluster boundary in a gene metric space $\mathfrak{G}_n^d$ is $O(k^2)$, where k is the size of the cluster.

Proof: Module (I) of this algorithm can run in $d \times k$ steps, and each of the other modules can run in d^2 steps. The above results are proven directly for modules (I) and (II), while for modules (III) and (IV) the proof uses proposition D.4. Hence, the order of one iteration of the algorithm is $O(k \times d)$. On the other hand, the number of iterations performed by the algorithm satisfies the inequality $k - (i-1)d \geq 0$, where

i is the number of iterations. Solving this inequality, we observe that the number of iterations is bounded by $\frac{k}{d} + 1$ and so, the order of ClusterBoundary is $O(k^2)$. $\square$

Bibliography

[1] M. B. Eisen, “Cluster 2.20 and treeview 1.60,” *Eisen Lab*, 2002. [Online]. Available: http://rana.lbl.gov/EisenSoftware.htm

[2] A. Saeed, V. Sharov, J. White, J. Li, W. Liang, N. Bhagabati, J. Braisted, M. Klapa, T. Currier, M. Thiagarajan, A. Sturn, M. Snuffin, A. Rezantsev, D. Popov, A. Ryltsov, E. Kostukovich, I. Borisovsky, Y. Liu, A. Vinsavich, V. Trush, and J. Quackenbush, “Tm4: a free, open-source system for microarray data management and analysis,” *Biotechniques*, vol. 34, pp. 374–378, 2003.

[3] M. Reich, K. Ohm, P. Tamayo, M. Angelo, and J. P. Mesirov, “Genecluster 2.0: An advanced toolset for bioarray analysis,” *Bioinformatics*, vol. 20, pp. 1797–1798, 2004.

[4] J. Seo and B. Shneiderman, “Interactively exploring hierarchical clustering results,” *Computer*, vol. 35, no. 7, pp. 80–86, 2002.

[5] G. H. Weber, O. Rübel, M.-Y. Huang, A. H. DePace, C. C. Fowlkes, S. V. Keränen, H. C. L. Luengo, H. Hagen, D. W. Knowles, J. Malik, M. D. Biggin, and B. Hamann, “Visual exploration of three-dimensional gene expression using physical views and linked abstract views,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 6, pp. 296–309, 2009.

[6] J. A. Castellanos-Garzón and F. Díaz, “An evolutionary computational model applied to cluster analysis of DNA microarray data,” *Submitted to Expert Systems with Applications, Elsevier*, 2012.

[7] ——, “clustergas: A hierarchical clustering method based on genetic algorithms,” CRAN R-Project. Department of Computer Science, University of Valladolid (Spain), http://cran.r-project.org/web/packages/clustergas, Tech. Rep., 2012, R package version 1.0.

[8] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, B. Marttine, Ed. Prentice Hall Englewood Cliffs, New Jersey 07632, 1998.

[9] A. K. Jain, N. M. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, september 1999.

[10] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data. An Introduction to Clustering Analysis.* John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.

[11] W. Pedrycz, *Knowledge-Based Clustering.* Published by John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.

[12] J. A. Castellanos-Garzón, C. A. García, P. Novais, and F. Díaz, "A visual analytics framework for cluster analysis of DNA microarray data," *Expert Systems with Applications, Elsevier*, vol. In Press - http://dx.doi.org/10.1016/j.eswa.2012.08.038, 2012.

[13] M. Schroeder, D. Gilbert, J. v. Helden, and P. Noy, "Approaches to vusualixation in bioinformatics: from dendrograms to Space Explorer," *Information Sciences, Elsevier*, vol. 139, pp. 19–57, 2001.

[14] D. A. Keim, "Information visualization and visual data mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, pp. 1–8, 2002.

[15] A. Inselberg and B. Dimsdale, "Parallel coordinates: a tool for visualizing multidimensional geometry," *VIS '90: Proceedings of the 1st conference on Visualization '90*, pp. 361–378, 1990.

[16] R Development Core Team, *R: A Language and Environment for Statistical Computing.*, R Foundation for Statistical Computing, Vienna, Austria, 2006, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org

[17] M. Affenzeller and R. Mayrhofer, "Generic heuristics for combinatorial optimization problems," *Proceedings of the 9th international conference on operational (KOI)*, pp. 83–92, 2002.

[18] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *SIGMOD 1998, Proc. ACM/SIGMOD Int'l Conf. Management of Data*, 1998, pp. 94–105.

[19] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms.* Addison-Wesley, Reading, Massachusetts, 1983.

[20] A. A. Alizadeh et al., "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, pp. 503–511, 2000.

[21] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences, USA*, vol. 96, pp. 6745–6750, june 1999.

[22] M. R. Anderberg, *Cluster Analysis for Applications.* Academic Press, Inc., New York, 1973.

[23] S. Arora and B. Barak, *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[24] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation, Combinatorial optimization problems and their approximability properties.* Springer Verlag, ISBN 3-540-65431-3, 2005.

[25] T. Baeck, G. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation.* U.K.: Institute of Physics, 1997.

[26] P. Baldin and S. Brunak, *Bioinformatics: The Machine Learning Approach.* MIT Press, Cambridge, MA, 1998.

[27] S. Bandyopadhyay and S. K. Kal, *Classification and Learning Using Genetic Algorithms, Applications in Bioinfomatics and Web Intelligence.* Springer Berlin Geidelberg, 2007.

[28] E. R. Banks, P. Agarwal, M. McBride, and C. Owens, "Toward a universal operator encoding for genetic programming," in *GECCO '09: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference.* New York, NY, USA: ACM, 2009, pp. 1983–1986.

[29] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and its Applications.* San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc., Jan. 1998.

[30] N. Belacel, Q. Wang, and M. Cuperlovic-Culf, "Clustering methods for microarray gene expression data." *OMICS: A Journal of Integrative Biology. A special issue on current microarray research*, vol. 10(4), pp. 507–531, 2006.

[31] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, "Discovering local structure in gene expression data: The order-preserving submatrix problem," in *Proc. Sixth Int'l Conf. Computational Biology (RECOMB '02)*, 2002, pp. 49–57.

[32] A. Ben-Dor, N. Friedman, and Z. Yakhini, "Class discovery in gene expression data," in *Proc. Fifth Ann. Int'l Conf. Computational Molecular Biology (RECOMB 2001)*, 2001, pp. 31–38.

[33] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering gene expression patterns," *J. Computational Biology*, vol. 6, pp. 281–297, 1999.

[34] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars, *Computational Geometry, Algorithms and Applications.* Springer-Verlag Berlin Heidelberg, Third Edition, ISBN 978-3-540-77973-5, 2008.

[35] D. P. Berrar, W. Dubitzky, and M. Granzow, *A Practical Approach to Microarray Data Analysis.* Kluwer Academic Publishers, New York, Boston, Dordrecht, London, Moscow, 2003.

[36] M. R. Berthold and L. O. Hall, "Visualizing fuzzy points in parallel coordinates," *Fuzzy Systems, IEEE Transactions on*, vol. 11, pp. 369–374, 2003.

[37] J. D. Boissonnat and M. Teillaud, *Mathematics and Visualization.* Springer-Verlag Berlin Heidelberg, 2006.

[38] P. E. Bourne and H. Wissig, *Structural Bioinformatics*, P. E. Bourne and H. Wissig, Eds. Wiley-Liss, Inc., Hoboken, New Jersey, 2003.

[39] A. Brazma, A. Robinson, G. Cameron, and M. Ashburner, "One-stop shop for microarray data," *Nature*, vol. 403, pp. 699–700, 2000.

[40] A. Brazma and J. Vilo, "Gene expression data analisis," *Federation of European Biochemical Societies Letters*, vol. 480, pp. 17–24, 2000.

[41] S. Busygin, G. Jacobsen, and E. Kramer, "Double conjugated clustering applied to leukemia microarray data," in *Proc. Second SIAM Int'l Conf. Data Mining, Workshop Clustering High Dimensional Data*, 2002.

[42] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, Inc., 1996.

[43] C. Böhm, A. Oswald, B. Richter C., Wackersreuther, and P. Wackersreuther, "Genetic algorithm for finding cluster hierarchies," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6860 LNCS (PART 1)*, pp. 349–363, 2011.

[44] A. Califano, G. Stolovitzky, and Y. Tu, "Analysis of gene expression microarays for phenotype classification," in *Conf. Computacional Molecular Biology*, 2000, pp. 75–85.

[45] J. W. Carmichael, J. A. George, and R. S. Julius, "Finding natural clusters," *Systematic Zoology*, vol. 17, pp. 144–150, 1968.

[46] J. A. Castellanos-Garzón, "Algoritmos genéticos para clustering de datos de expresión génica," Master's thesis, Masters Thesis, Department of Computer Science and Automatics, University of Salamanca, Spain, 2006.

[47] J. A. Castellanos-Garzón and L. A. Miguel-Quintales, "Un método de clustering jerárquico basado en algoritmos genéticos," *II Simposio de Inteligencia Computacional (SICO), II Congreso Español de Informática (CEDI), IEEE Computational Intelligence Society (SC), Thomson Editores, ISBN: 978-84-9732-606-3*, vol. S15, pp. 177–185, 2007.

[48] ——, "Un método de clustering jerárquico con algoritmos genéticos: un caso práctico," *I Workshop Español sobre Extracción y Validación de conocimiento en Bases de Datos Biomédicas (EVaBio'07), CAEPIA, ISBN-13:978-84-611-8854-3*, vol. W8, pp. 1–11, 2007.

[49] Z. S. H. Chan and N. Kasabov, "Gene trajectory clustering with a hybrid genetic algorithm and expectation maximization method," *IEEE International Joint Conference on Neural Networks*, vol. 3, pp. 1669–1674, july 2004.

[50] G. Chen, S. A. Jaradat, N. Banerjee, T. S. Tanaka, M. S. Ko, and M. Q. Zhang, *Evaluation and Comparison of Clustering Algorithms in Analyzing ES Cell Gene Expression Data.* Statistica Sinica, 2002, vol. 12.

[51] Q. Chen, J. Han, Y. Lai, W. He, and K. Mao, "Clustering problem using adaptive genetic algorithm," *Advances in Natural Computation, Springer Berlin / Heidelberg*, vol. 3612, pp. 782–786, 2005.

[52] Y. Cheng and G. Church, "Biclustering of expression data," in *Proc. Eighth Int'l Conf. Intelligent Systems for Molecular Biology (ISMB ' 00)*, 2000, pp. 93–103.

[53] W. Chih-Ping, L. Yen-Hsien, and H. Che-Ming, “Empirical comparison of fast clustering algorithms for large data sets.” *Proceedings of the 33rd Hawaii International Conference on System Sciences. IEEE*, pp. 1–10, 2000.

[54] H. Chipman, T. Hastie, and R. Tibshirani, “Clustering microarray data,” *Statistical Analysis of Gene Expression Microarray Data*, 2003.

[55] H. Chipman and R. Tibshirani, “Hybrid hierarchical clustering with applications to microarray data,” *Biostatistics*, vol. 7, pp. 302–317, 2006.

[56] H. Chipman, R. Tibshirani, and with tsvq code originally from Trevor Hastie, *hybridHclust: Hybrid hierarchical clustering*, 2006, R package version 1.0-1. [Online]. Available: http://ace.acadiau.ca/math/chipmanh/hybridHclust

[57] H. Cho, I. Dhillon, Y. Guan, and S. Sra, “Minimum sum-squared residue cococlustering of gene expression data,” in *Proc. Fourth SIAM Int’l Conf. Data Mining*, 2004.

[58] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis, “A genome-wide transcriptional analysis ofthe mitotic cell cycle,” *Mol. Cell*, vol. 2, pp. 65–73, 1998.

[59] P. C. Chu and J. E. Beasley, “A genetic algorithm for the set partitioning problem,” *Technical report, Imperial College, The Management School, London, England*, pp. 481–487, 1995.

[60] F. A. Cleveland and S. F. Smith, “Using genetic algorithms to schedule flow shop releases.” in *Proceedings of the 3rd International Conference on Genetic Algorithms*, 1989, pp. 160–169.

[61] W. J. Conover, *Practical Nonparametric Statistics.* John Wiley & Sons, Inc ., New York, 1971.

[62] E. P. Consortium, “Identification and analysis of functional elements in 1% of the human genome by the encode pilot project,” *Nature*, vol. 447, pp. 799–816, 2007.

[63] ——, “The encode (encyclopedia of DNA elements) project,” *Science*, vol. 306, pp. 636–640, 2004.

[64] S. A. Cook, “The complexity of theorem-proving procedures,” in *STOC ’71: Proceedings of the third annual ACM symposium on Theory of computing.* New York, NY, USA: ACM, 1971, pp. 151–158.

[65] P. Crescenzi and V. Kann, *A compendium of NP optimization problems.* http://www.nada.kth.se/ viggo/wwwcompendium/wwwcompendium.html, 2005.

[66] P. Dh́aeseleer, X. Wen, S. Fuhrman, and R. Somogyi, "Mining the gene expression matrix: Inferring gene relationships from large scale gene expression data," *Information Processing in Cells and Tissues*, pp. 203–212, 1998.

[67] S. Datta and S. Datta, "Comparisons and validation of statistical clustering techniques for microarray gene expression data," *Bioinformatics, Oxford University*, vol. 19, pp. 459–465, 2003.

[68] K. A. De Jong and W. M. Spears, "Using genetic algorithms to solve NP-complete problems," in *Proceedings of the third international conference on Genetic algorithms*, 1989.

[69] K. De-Jong, "An analysis of the behaviour of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, Ann Arbor, 1975.

[70] K. A. De-Jong and W. M. Spears, "Using genetic algorithms to solve np-complete problems," *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 124–132, 1989.

[71] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, J. D. Schaffer, Ed., 1989, pp. 42–50.

[72] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2002.

[73] K. A. DeJong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, Ann Arbour. Department of Computer and Communication Sciences, 1975.

[74] J. DeRisi, V. Iyer, and P. Brown, "Exploring the metabolic and genetic control of gene expression on a genomic scale," *Science*, vol. 278 (5338), pp. 680–686, 1997.

[75] J. DeRisi, L. Penland, P. Brown, P. Bittner, M.L.and Meltzer, M. Ray, Y. Chen, Y. Su, and J. Trent, "Use of a cDNA microarray to analyse gene expression patterns in human cancer," *Nature Genetics*, vol. 14 (4), pp. 457–460, 1996.

[76] K. Devlin, *The Millennium Problems. The Seven Greatest Unsolved Mathematical Puzzles of Our Time*, K. Devlin, Ed. Basic Books, A Member of the Perseus Books Group, 2002.

[77] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[78] Q. Ding and J. Gasvoda, "A genetic algorithms for clustering on image data," *International Journal of Computational Intelligence*, vol. 1, pp. 75–80, 2004.

[79] M. Dorigo and T. stützle, *Ant Colony Optimization.* The MIT Press, 2004.

[80] M. Dorigo and G. D. Caro, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, pp. 137–172, 1999.

[81] M. Dorigo and L. Gambardella, "Ant colon system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. on Evolutionary Computation*, vol. 1, pp. 53–66, 1997.

[82] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification (2nd edition).* Wiley, New York, 2001.

[83] H. A. Edelstein, *Introduction to Data Mining and Knowledge Discovery.* Two Crows Corporation (Third Ed.), 2005.

[84] B. Efron, "The jackknife, the bootstrap, and other resampling plans," *Proc. CBMS-NSF Regional Conf. Series in Applied Math.*, vol. 38, 1982.

[85] A. E. Eiben, "Multiparent recombination in evolutionary computing," in *Advances in evolutionary computing: theory and applications*, ser. Natural Computing Series. Springer-Verlag New York, Inc., 2003, pp. 175–192.

[86] M. Eisen, T. Spellman, P. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences, USA*, vol. 95, pp. 14 863–14 868, 1998.

[87] F. Emmert-Streib and M. Dehmer, *Analysis of Microaray Data. A Networking-Based Approach.* Whiley-VCH Verlag GmbH & Co. kGaA, 2008.

[88] A. P. Engelbrecht, *Computational Intelligence. An Introduction.* John Wiley & Sons, Ltd (2nd ed.), 2007.

[89] L. J. Eshelman, "The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination," in *Foundations of Genetic Algorithms-1*, G. J. E. Rawlins, Ed., 1991, pp. 265–283.

[90] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996.

[91] B. S. Everitt, *Cluster Analysis.* John Wiley & Sons. Inc., New York, 1974.

[92] V. Fernández, B. R. García, and R. R. L. González, "Genetic algorithms applied to clustering," *Proceedings of international conference on signal and image processing*, pp. 97–99, 1996.

[93] I. K. Fodor, "A survey of dimension reduction techniques," Lawrence Livermore National Laboratory (LLNL), UCRL-ID-148494, University of California, Livermore, CA, Tech. Rep., 2002.

[94] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, pp. 1–16, 1995.

[95] C. Fraley and A. Raftery, "How many clusters? which clustering method? answers via model-based cluster analysis," *The Computer J.*, vol. 41, pp. 578–588, 1998.

[96] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner, "Structure-based brushes: A mechanism for navigating hierarchically organized data and information spaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 2, pp. 150–159, Apr./Jun. 2000.

[97] P. Gács and L. Lovász, *Complexity of Algorithms.* Lecture Notes,Springer-Verlag Berlin Heidelberg, 1999.

[98] J. Gao, P. W. Kwan, and Y. Guo, "Robust multivariate L1 principal component analysis and dimensionality reduction," *Neurocomputing, Elsevier*, vol. 72, pp. 1242–1249, 2009.

[99] M. E. e. a. Garber, "Diversity of gene expression in adenocarcinoma of the lung," *Proceedings of the National Academy of Sciences*, vol. 98, pp. 13 784–13 789, 2001.

[100] J. M. Geoffrey, K. A. Do, and C. Ambroise, *Analyzing Microarray Gene Expression Data.* John Wiley & Sons, Inc., Hoboken, New Jersey, 2004.

[101] A. Gersho and R. Gray, "Vector quantization and signal compression," in *Boston: Kluwer Academic Publishers*, 1992.

[102] G. Getz, E. Levine, and E. Domany, "Coupled two-way clustering analysis of gene microarray data," in *Proc. Natural Academy of Sciences US*, 2000, pp. 12 079–12 084.

[103] D. Ghosh and A. Chinnaiyan, "Mixture modelling of gene expression data from microarray experiments," *Bioinformatics*, vol. 18, pp. 275–286, 2002.

[104] P. Godefriud and S. Khurshid, "Exploring very large state spaces using genetic algorithms," *Springer-Verlag Berlin Heidelberg*, vol. 2280, pp. 266–280, 2002.

[105] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley Longman, Inc., 1989.

[106] D. E. Goldberg and J. J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd International Conference on Genetic Algorithms*, 1987, pp. 41–49.

[107] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gassenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, D. Bloomfield, and E. Lander, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, pp. 531–537, 1999.

[108] W. A. Greene, "Unsupervised hierarchical clustering via a genetic algorithm," *Congress on Evolutionary Computation, CEC'03 IEEE*, vol. 2, pp. 998–1005, 2003.

[109] ——, "Genetic algorithms for partitioning sets," *International Journal in Artirficial Intelligence Tools, Word Scientific Publishing Company, Singapore*, vol. 10, pp. 225–241, 2001.

[110] J. F. Hair, B. Black, B. Babin, R. E. Anderson, and R. L. Tatham, *Multivariate Data Analysis.* London: Prentice Hall, 6th Edition, 2005.

[111] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," in *Intelligent Information Systems J.*, 2001.

[112] A. V. Hall, "Group forming and discrimination with homogeneity functions," in *Numerical Taxonomy (A. J. Cole, ed.), Academic Press, Inc., New York*, 1969, pp. 53–67.

[113] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, J. Gray, Ed. Elsevier Inc., 2006.

[114] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining.* The MIT Press, 2001.

[115] J. Handl, J. Knowles, and D. B. Kell, "Computational cluster validation in post-genomic data analysis," *Published by Oxford University Press*, vol. 21, pp. 3201–3212, 2005.

[116] J. Hansohm, "Two-mode clustering with genetic algorithms," *Classification, Automation, and New Media: Proceedings of the 24th Annual Conference of the Gesellschaft Fur Klassifikation E.V.*, pp. 87–94, 2000.

[117] J. Hartigan, "Direct clustering of a data matrix," *J. Am. Statistical Assoc. (JASA)*, vol. 67, pp. 123–129, 1972.

[118] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms.* Published by John Wiley & Sons, Inc., Hoboken, New Jersey, Second Edition, 2004.

[119] B. Heckel, B. Hamann, and A. E. Uva, "Cluster-based generation of hierarchical surface models," *Scientific Visualization Conference, IEEE Computer Society*, pp. 105–114, 1997. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/DAGSTUHL.1997.10036

[120] L. J. Heyer, S. Kruglyak, and S. Yooseph, "Exploring expression data: Identification and analysis of coexpressed genes," *Genome Research*, vol. 9, pp. 1106–1115, 1999.

[121] S. Hiley, J. Jackman, T. Babak, M. Trochesset, Q. Morris, E. Phizicky, and T. Hughes, "Detection and discovery of rna modifications using microarrays," *Nucleic Acids Research*, vol. 733 (1), p. e2, 2005.

[122] A. Hill, E. Brown, M. Whitley, G. Tucker-Kellogg, C. Hunter, and D. Slonim, "Evaluation of normalization procedures for oligonucleotide array data based on spiked cRNA contros," *Genome Biology*, vol. 2, pp. research0055.–1–0055.13, 2001.

[123] J. Hoheisel, "Microarray technology: beyond transcript profiling and genotype analysis," *Nature Reviews Genetics*, vol. 7 (3), pp. 200–210, 2006.

[124] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* MIT Press Edition, 1992.

[125] H. Hoppe, "Surface reconstruction from unorganized points," Ph.D. dissertation, University of Washington, Department of Computer Science and Engineering, 1994.

[126] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise smooth surface reconstruction," in *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques.* New York, NY, USA: ACM, 1994, pp. 295–302.

[127] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques.* New York, NY, USA: ACM, 1992, pp. 71–78.

[128] E. R. Hruschka, R. J. G. B. Campello, and N. Castro, "Evolving clusters in gene-expression data," *Information sciences, An international journal, Elsevier*, vol. 176, pp. 1898–1927, 2006.

[129] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. Leon F. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 39, pp. 133–155, 2009.

[130] E. R. Hruschka, L. N. de Castro, and R. J. G. B. Compello, "Evolutionary algorithms for clustering gene-expression data," *Proceedings of the fourth IEEE International Conferece on Data Mining*, pp. 403–406, 2004.

[131] L. J. Hubert, "Some applications of graph theory to clustering," *Psychometrika*, vol. 39, pp. 283–309, 1974.

[132] C. J. Huberty, *Applied Discriminant Analysis.* John Wiley & Sons, Inc., New York, 1994.

[133] T. Hughes, M. Marton, A. Jones, C. Roberts, R. Stoughton, C. Armour, H. Bennett, E. Coffey, H. Dai, Y. He, M. Kidd, A. King, M. Meyer, D. Slade, P. Lum, S. Stepaniants, D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. Friend, "Functional discovery via a compendium of expression profiles," *Cell*, vol. 102 (1), pp. 109–126, 2000.

[134] L. Jakt, L. Cao, K. Cheah, and D. Smith, “Assessing clusters and motifs from gene expression data,” *Genome Research*, vol. 11, pp. 1112–1123, 2001.

[135] D. Jiang, J. Pei, and A. Zhang, “DHC: A density-based hierarchical clustering method for time series gene expression data,” in *Proceedings of the Third IEEE Symposium on BioInformatics and BioEngineering (BIBE)*, 2003.

[136] ——, “Interactive exploration of coherent patterns in time-series gene expression data,” in *Proc. Ninth ACM SIGKDD Intĺ Conf. Knowledge Discovery and Data Mining (SIGKDD0́3)*, 2003.

[137] D. Jiang, C. Tang, and A. Zhang, “Cluster analysis for gene expression data: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1370–1386, 2004.

[138] G. Johannes, M. Carter, M. Eisen, P. Brown, and P. Sarnow, “Identification of eukaryotic mRNAs that are translated at reduced cap binding complex eIF4F concentrations using a cDNA microarray,” *Proceedings of theNational Academy of Sciences of the United States of America*, vol. 96 (23), pp. 13 118–131 123, 1999.

[139] J. Johansson, R. Treloar, and M. Jern, “Integration of unsupervised clustering, interaction and parallel coordinates for the exploration of large multivariate data,” in *IV ’04, Proceedings of the Information Visualisation, Eighth International Conference, IEEE Computer Society.* Washington, DC, USA: IEEE Computer Society, 2004, pp. 52–57.

[140] D. S. Johnson, “A catalog of complexity classes,” *Handbook of theoretical computer science (vol. A): algorithms and complexity, MIT Press Cambridge, MA, USA*, pp. 67–161, 1991.

[141] I. Jolliffe, *Principal Component Analysis, Second Edition*, N. Y. I. Springer-Verlag, Ed. Spriger Series in Statistics, 2002.

[142] I. T. Jolliffe, *Principal Component Analysis.* Springer-Verlag, 2002.

[143] L. Jourdan, C. Dhaenens, and E. T. Ghazali, “A genetic algorithm to exploit genetic data,” *D. Corne and G. Fogel editor, Evolutionary Computation in Bioinformatics, Morgan Kaufmann*, 2002. [Online]. Available: citeseer.ist.psu.edu/717663.html

[144] N. Kambhatla and L. T. K., “Dimension reduction by local principal component analysis,” *Neural Computation*, vol. 9, pp. 1493–1516, 1997.

[145] E. Keedwell and A. Narayanan, "Genetic algorithms for gene expression analysis," *Lecture Notes in Computes Science Springer-Verlag GmbH EvoWorkshops*, vol. 2611, pp. 76–86, 2003.

[146] D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler, "Challenges in visual data analysis," in *IV '06, Proceedings of the conference on Information Visualization.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 9–16.

[147] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon, "Visual analytics: Definition, process, and challenges," in *Information Visualization. Human-Centered Issues and Perspectives.* Springer, 2008, pp. 154–175. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-70956-5_7

[148] M. G. Kendall, "Discrimination and classification," in *Multivariate Analysis (P. R. Krishnaiah, ed.), Academic Press, Inc., New York*, 1966, pp. 165–185.

[149] A. Khodursky, B. Peter, M. Schmid, J. DeRisi, D. Botstein, P. Brown, and N. Cozzarelli, "Analysis of topoisomerase function in bacterial replication fork movement: use of DNA microarrays," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 97 (17), pp. 9419–9424, 2000.

[150] K. E. Kinnear, *Advances in Genetic Programming*, K. E. Kinnear, Ed. Massachusetts Institute of Technology, ISBN 0-261-11188-8, 1994.

[151] Y. Klugar, R. Basri, J. Chang, and M. Gerstein, "Spectral biclustering of microarray data: Coclustering genes and conditions," *Genome Research*, vol. 13, pp. 703–716, 2003.

[152] T. Kohonen, *Self-Organization and Associative Memory.* Berlin: Spring-Verlag, 1984.

[153] P. G. Kolaitis and M. N. Thakur, "Logical definability of np optimization problems," *Inform. and Comput.*, vol. 115, pp. 321–353, 1994.

[154] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries," in *Proc. ACM SIGMOD*, 2000, pp. 201–212.

[155] B. Korte and J. Vygen, *Combinatorial Optimization. Theory and Algorithms.* Springer-Verlag Berlin Heidelberg, 2008.

[156] ——, "DHC: A density-based hierarchical clustering method for time series gene expression data," in *Proceedings of the Third IEEE Symposium on BioInformatics and BioEngineering (BIBE)*, 2003.

[157] J. Koza, F. Bennett, D. Andre, and M. Keane, "Evolutionary design of analog electrical circuits using genetic programming," in *Proceedings of Adaptive Computing in Design and Manufacture Conference*, April 1998, pp. 21–23.

[158] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA:MIT Press, 1992.

[159] S. G. Krantz, D. Saltman, S. D., and S. R., *Metric Spaces.* Library of Congress Cataloging-in-Publication Data, 1964.

[160] N. Lama, P. Boracchi, and E. Biganzoli, "Benchmarking information discovery methods for microarray data in breast cancer," in *European Conferece on EACDA*, 2005. [Online]. Available: http://ciml.di.unipi.it/EACDA2005/papers.html

[161] G. N. Lance and W. T. Williams, "A general theory of classificatory sorting strategics: II. clustering systems," *Computer Journal*, vol. 10, pp. 271–277, 1982.

[162] W. B. Langdon and R. Poli, *Fundation of Genetic Programming.* ISBN-3-540-42451-2, Springer-Verlag, 1998.

[163] D. T. Larose, *Data mining methods and models.* John Wiley & Sons, Inc., 2006.

[164] M. Laszlo and S. Mukherjee, "A genetic algorithm that exchanges neighboring centers for k-means clustering," *Pattern Recognition Letture*, vol. 28, pp. 2359–2366, 2007.

[165] L. Lazzeroni and A. Owen, "Plaid models for gene expression data," Stanford University, Tech. Rep., 2000.

[166] J. Liu and W. Wang, "OP-cluster: Clustering by tendency in high dimensional space," in *Proc. Third IEEE Int'l Conf. Data Mining*, 2003, pp. 187–194.

[167] X. Llorá, K. Sastry, F. Alías, D. E. Goldberg, and M. l. Welge, "Analyzing active interactive genetic algorithms using visual analytics," in *GECCO '06, Proceedings of the 8th annual conference on Genetic and evolutionary computation.* New York, NY, USA: ACM, 2006, pp. 1417–1418.

[168] D. Lockhart, "Expression monitoring by hybridization to high-density oligonucleotide arrays," *Nature Biotechnology*, vol. 14, pp. 1675–1688, 1996.

[169] J. A. Lozano and P. Larrañaga, "Applying genetic algorithms to search for the best hierarchical clustering of a dataset," *Pattern Recognittion Letters*, vol. 20, pp. 911–918, 1999.

[170] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming.* Springer, 3rd ed., 2008.

[171] V. Lumelsky, "A combined algorithm for weighting the variables and clustering in the clustering problem," *Pattern Recognition*, vol. 15, pp. 53–60, 1982.

[172] R. Mägi, A. Pfeufer, M. Nelis, A. Montpetit, A. Metspalu, and M. Remm, "Evaluating the performance of commercial whole-genome marker sets for capturing common genetic variation," *BMC Genomics*, vol. 118, p. 159, 2007.

[173] P. C. H. Ma, K. C. C. Chan, X. Yao, and D. K. Y. Chiu, "An evolutionary clustering algorithm for gene expression microarray data analysis," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 296–314, 2006.

[174] P. Macnaughton-Smith, W. T. Williams, M. B. Dale, and L. G. Mockett, "Dissimilarity analysis: a new technique of hierarchical subdivision," *Nature*, vol. 202, pp. 1034–1035, 1965.

[175] S. C. Madeira and A. L. Oliveira, "Biclustering algorithms for biological data analysis: A survey," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 1, pp. 24–44, 2004.

[176] M. Maechler, based on S original by P. Rousseeuw, Anja.Struyf@uia.ua.ac.be, Mia.Hubert@uia.ua.ac.be, and initial R port by Kurt.Hornik@R-project.org, *cluster: Cluster Analysis Extended Rousseeuw et al.*, 2005, R package version 1.10.2.

[177] J. I. Maletic, A. Marcus, and M. L. Collard, "A task oriented view of software visualization," *IEEE Workshop of Visualizing Software for Understanding and Analysis (VISSOFT)*, vol. 26, pp. 32–40, 2002.

[178] M. A. Maloof, *Machine Learning and Data Mining for Computer Security. Methods and Applications.* Springer-Verlag, 2006.

[179] M. Marić, F. Marić, Z. Mijajlović, and B. Jovanović, "Automatic construction of surface model," School of Mathematics, University of Belgrade, Belgrade, Serbia and Montenegro, Tech. Rep., 2005.

[180] U. Maulik and S. Bandyopadhyay, "Genetic algorithms-based clustering technique," *The Journal of the Pattern Recognition Society Published by Elsevier Science Ltd.*, vol. 33, pp. 1455–1465, 2000.

[181] D. Mazur, "Clustering based on genetic algorithms," *Artificial Intelligence Methods (AI-METH) Gliwice, Poland*, November 17-19 2004.

[182] J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, *Evolutionary Programming.* Proceedings of the Fourth Annual Conference on Evolutionary Programming, Massachusetts Institute of Technology, 1995.

[183] G. McLachlan, R. Bean, and D. Peel, "A mixture model-based approach to the clustering of microarray expression data," *Bioinformatics*, vol. 18, pp. 413–422, 2002.

[184] J. McQueen, "Some methods for classification and analysis of multivariate observations," *Proc. Fifth Berkeley Symp. Math. Statistics and Probability*, vol. 1, pp. 281–297, 1967.

[185] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, New York, Third Edition, 1999.

[186] M. Mitchell, *An Introduction to Genetic Algorithms.* MIT Press, 1998.

[187] E. Moler, M. Chow, and I. Mian, "Analysis of molecular profile data using generative and discriminative methods," *Physiological Genomics*, vol. 4, pp. 109–126, 2002.

[188] J. Nocedal and S. J. Wright, *Numerical Optimization*, P. Glynn and S. M. Robinson, Eds. Springer Series in Operations Research, 1999.

[189] V. Olman, F. Mao, H. Wu, and Y. Xu, "Parallel clustering algorithm for large data sets with applications in bioinformatics," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 6, pp. 344–352, 2009.

[190] D. L. Olson and D. Delen, *Advanced Data Mining Techniques.* Springer-Verlag Berlin Heidelberg, 2008.

[191] S. K. Pal, S. Bandyopadhyay, and C. A. Murthy, "Evolutionary computation in bioinformatics: A review," *IEEE Transactions on Systems, Man and Cybernerics, Part C*, vol. 36, pp. 601–615, 2006.

[192] S. K. Pal and S. C. K. Shiu, *Foundations of Soft Case-Based Reasoning*, J. S. Albus, A. M. Meyste, and L. A. Zadeh, Eds. John Wiley & Sons, Inc., Hoboken, New Jersey, 2004.

[193] C. H. Papadimitriou, *Computational Complexity.* Addison-Wesley, Reading MA, 1994.

[194] C. H. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," *J. Comput. System Sci.*, vol. 43, pp. 425–440, 1991.

[195] R. Parker and U. Sheth, "P bodies and the control ofmrnatranslation and degradation," *Molecules and Cells*, vol. 25 (5), pp. 635–646, 2007.

[196] D. Peña, *Análisis de datos multivariantes*, C. F. Madrid, Ed. McGRAW-HILL, IBN: 84-481-3610-1, 2002.

[197] J. Pollack, C. Perou, A. Alizadeh, M. Eisen, A. Pergamenschikov, C. Williams, S. Jeffrey, D. Botstein, and P. Brown, "Genome-wide analysis of dna copy-number changes using cDNA microarrays," *Nature Genetics*, vol. 23 (1), pp. 41–46, 1999.

[198] C. Poloni and V. Pediroda, "GA coupled with computationally expensive simulation: Tools to improve efficiency," in *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, D. Quagliarella, J. Periaux, C. Poloni, and G. Winter, Eds. John-Wiley and Sons, Sussex, England, 1997, pp. 267–288.

[199] A. S. Pérez, *Una Introducción a la Computación Evolutiva.* http://neo.lcc.uma.es/EAWebSite/web/ EvolutionaryComputation.html, 1996.

[200] P. Ralf-Herwig, C. Muller, C. Bull, H. Lehrach, and J. OBrien, "Large-scale clustering of cDNA-fingerprinting data," *Genome Research*, vol. 9, pp. 1093–1105, 1999.

[201] K. Ramanathan and S. Uei-Guan, "Clustering and combinatorial optimization in recursive supervised learning," *Journal of Combinatorial Optimization, Springer Netherlands*, vol. 13, pp. 137–152, 2006.

[202] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, pp. 846–850, 1971.

[203] C. Rasmussen, B. de la Cruz, Z. Ghahramani, and D. Wild, "Modeling and visualizing uncertainty in gene expression clusters using dirichlet process mixtures," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 6, pp. 615–628, 2009.

[204] I. Rechenberg, "Evolution strategy," in *Zuarda et.al.*, 1994, pp. 147–159.

[205] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proc. IEEE*, vol. 96, pp. 2210–2239, 1998.

[206] K. Rose, E. Gurewitz, and G. Fox, "Statistical mechanics and phase transitions in clustering," *Physical Rev. Letters*, vol. 65, pp. 945–948, 1990.

[207] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[208] O. P. Rud, *Data Mining Cookbook. Modeling Data for Marketing, Risk, and Customer Relationship Management*, R. M. Elliott, Ed. John Wiley & Sons, Inc., 2001.

[209] O. Rübel, G. H. Weber, M.-Y. Huang, E. W. Bethel, M. D. Biggin, C. C. Fowlkes, H. C. L. Luengo, S. V. E. Keränen, M. Eisen, D. Knowles, J. Malik, H. Hagen, and B. Hamann, "Integrating data clustering and visualization for the analysis of 3d gene expression data," *IEEE Transactions on Computational Biology and Bioinformatics*, 2008, lBNL-382E, to appear.

[210] M. Salicru, S. Vives, and T. Zheng, "Inferential clustering approach for microarray experiments with replicated measurements," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 6, pp. 594–604, 2008.

[211] S. Salvador and P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms," in *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2004.

[212] M. D. Schena, R. Shalon, R. Davis, and P. Brown, "Quantitative monitoring of gene expression patterns with a complementary DNA microarray," *Science*, vol. 270, pp. 467–470, 1995.

[213] E. Segal, A. Battle, and D. Koller, "Decomposing gene expression into cellular processes," in *Proc. Pacific Symp. Biocomputing*, vol. 8, 2003, pp. 89–100.

[214] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller, "Rich probabilistic models for gene expression," *Bioinformatics*, vol. 17, pp. S243–S252, 2001.

[215] J. Seo and B. Shneiderman, "A knowledge integration framework for information visualization," *From Integrated Publication and Information Systems to Information and Knowledge Environments, LNCS, Springer Berlin/Heidelberg*, vol. 3379/2005, pp. 207–220, 2005.

[216] A. Shadeo and W. Lam, "Comprehensive copy number profiles of breast cancer cell model genomes," *Breast Cancer Research*, vol. 8 (1), p. R9, 2006.

[217] R. Shamir and R. Sharan, “Click: A clustering algorithm for gene expression analysis,” *In Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB ’00). AAAI Press*, 2000.

[218] G. Sherlock, “Analysis of large-scale gene expression data,” *Current Opinion in Immunology*, vol. 12, pp. 201–205, 2000.

[219] J. Shlens, “A tutorial on principal component analysis, institute for nonlinear science, university of california, san diego,” Systems Neurobiology Laboratory, Salk Insitute for Biological Studies, Tech. Rep., 2005.

[220] B. Shneiderman, “The eyes have it: A task by data type taxonomy for information visualizations,” *IEEE Symposium on Visual Languages*, vol. 0, p. 336, 1996.

[221] G. F. Simmons, *Introduction to Topology and Modern analysis.* McGRA W-H1lL BOOK COMPANY, INC., 1963.

[222] S. Sivanandam and S. Deepa, *Introduction to Genetic Algorithms.* Springer-Verlag Berlin Heidelberg, 2008.

[223] F. Smet, J. Mathys, K. Marchal, G. Thijs, M. Moor, D. Bart, and Y. Moreau, “Adaptive quality-based clustering of gene expression profiles,” *Bioinformatics*, vol. 18, pp. 735–746, 2002.

[224] R. Smith, S. Forrest, and A. S. Perelson, “Population diversity in an immune system model: Implications for genetic search,” in *Foundations of Genetic Algorithms 2.* Kaufmann, M., 1992, pp. 153–166.

[225] R. R. Sokal, *Clustering and Classification: Background and Current Directions.* Classification and Clustering, Academic Press, 1977.

[226] T. e. a. Sorlie, “Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications,” *PNAS*, vol. 98, pp. 10 969–74, 2001.

[227] T. Speed, *Statistical Analysis of Gene Expression Microarray Data*, T. Speed, Ed. Chapman & Hall/CRC Press LLC, 2003.

[228] F. Spitzer, *Principles of Random Walk.* D. Van Nostrand, Princeton, 1964.

[229] M. Srinivas and L. M. Patnaik, “Genetic algorithms: A survey,” *Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[230] P. Takizawa, J. DeRisi, J. Wilhelm, and R. Vale, "Plasma membrane compartmentalization in yeast by messenger rna transport and a septin diffusion barrier," *Science*, vol. 290 (5490), pp. 341–344, 2000.

[231] P. Tamayo, D. Solni, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. Lander, and T. Golub, "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation," *Proc. Natĺ Academy of Science*, vol. 96, pp. 2907–2912, 1999.

[232] A. Tanay, R. Sharan, and R. Shamir, "Discovering statistically significant biclusters in gene expression data," *Bioinformatics*, vol. 18, pp. S136–S144, 2001.

[233] C. Tang, A. Zhang, and J. Pei, "Mining phenotypes and informative genes from gene expression data," in *Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD'03)*, 2003.

[234] C. Tang, L. Zhang, and A. Zhang, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms," in *Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS). IEEE*, 2002.

[235] C. Tang, L. Zhang, I. Zhang, and M. Ramanathan, "Interrelated two-way clustering: An unsupervised approach for gene expression data analysis," in *Proc. Second IEEE Int' l Symp. Bioinformatics and Bioeng*, 2001, pp. 41–48.

[236] Y. Tao, D. Papadias, and X. Lian, "Reverse KNN search in arbitrary dimensionality," in *Proc. Int'l Conf. Very Large Data Bases*, 2004, pp. 744–755.

[237] S. Tavazoie, D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church, "Systematic determination of genetic network architecture," *Nature Genetics*, pp. 281–285, 1999.

[238] A. Tefferi, E. Bolander, M. Ansell, D. Wieben, and C. Spelsberg, "Primer on medical genomics part III: Microarray experiments and data analysis," *Mayo Clinic Proc.*, vol. 77, pp. 927–940, 2002.

[239] J. Thomas and K. Cook, "A visual analytics agenda," *Computer Graphics and Applications, IEEE*, vol. 26, no. 1, pp. 10–13, Jan.-Feb. 2006.

[240] J. Thomas and K. Cook., *Illuminating the Path: Research and Development Agenda for Visual Analytics.* IEEE-Press, 2005.

[241] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. Altman, “Missing value estimation methods for dna microarrays,” *Bioinformatics*, vol. 6, pp. 520–5, 2001.

[242] W. Visser, *The Cognitive Artifacts of Designing.* Lawrence Erlbaum Associates, Inc., Mahwah, New Jersey, 2006.

[243] S. Wagner, M. Affezeller, and D. Schragl, “Traps and dangers when modelling problems for genetic algorithms.” *Austrian Society for Cybernetic Studies*, pp. 79–84, 2004.

[244] R. S. Wang, Y. Wang, X. S. Zhang, and L. Chen, “Inferring transcriptional regulatory networks from high-throughput data,” *Bioinformatics, Oxford University Press*, pp. 1–8, 2007.

[245] X. Wang and K. K. Paliwal, “Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition,” *Pattern Recognition, Elsevier*, vol. 36, pp. 2429–2439, 2003.

[246] E. J. Wegman and Q. Luo, “High dimensional clustering using parallel coordinates and the grand tour,” *Computing Science and Statistics*, vol. 28, pp. 361–368, 1996.

[247] H. S. Wilf, *Algorithms and Complexity.* Internet Edition, 1994. [Online]. Available: http://www.cscis.upenn.edu/wilf

[248] S. S. Wilks, *Mathemmical Statistics.* John Wiley & Sons, Inc., New York, 1963.

[249] E. Winzeler, D. Richards, A. Conway, A. Goldstein, S. Kalman, M. McCullough, J. McCusker, D. Stevens, L. Wodicka, D. Lockhart, and R. Davis, “Direct allelic variation scanning of the yeast genome,” *Science*, vol. 281 (5380), pp. 1194–1197, 1998.

[250] C. Xia, W. Hsu, M. L. Lee, and B. C. Ooi, “Border: Efficient computation of boundary points,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 289–303, 2006.

[251] E. Xing and R. Karp, “Cliff: Clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts,” *Bioinformatics*, vol. 17, pp. 306–315, 2001.

[252] J. Yang, W. Wang, H. Wang, and P. Yu, “Enhanced biclustering on expression data,” in *Proc. Third IEEE Conf. Bioinformatics and Bioeng.*, 2003, pp. 321–327.

[253] ——, “d-clusters: Capturing subspace correlation in a large data set,” in *Proc. 18th IEEE Int’l Conf. Data Eng.*, 2002, pp. 517–528.

[254] K. Yee-Yeung, “Clustering analysis of gene expression data,” Ph.D. dissertation, University of Washintong, 2001.

[255] K. Yeung, C. Fraley, A. Murua, A. Raftery, and W. Ruzz, “Model-based clustering and data transformations for gene expression data,” *Bioinformatics*, vol. 17, pp. 977–987, 2001.

[256] K. Y. Yeung, D. R. Haynor, and W. L. Ruzzo, “Validating clustering for gene expression data,” *Bioinformatics, Oxford University*, vol. 17, pp. 309–318, 2001.

[257] J. Zhang, H. Shu-Hung-Chung, and W. L. Lo, “Clustering-based adaptive crossover and mutation probabilities for genetic algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 326–335, 2007.

[258] Y. Zhou and J. He, “A runtime analysis of evolutionary algorithms for constrained optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 608–619, 2007.