

Looking for high density areas in a graph

Application to orthologous genes

Tristan Colombo* Alain Guénoche† Yves Quentin‡

Abstract : *The aim of this paper is to introduce new methods to build classes of vertices in a graph. These classes correspond to connected parts having a density of inner edges which is higher than the average on the whole graph. They are progressively determined; a kernel of each class is first established, and it is extended to connected elements; several strategies are compared. A validation of this graph partitioning method is made with random graphs fulfilling some density conditions. An experimental study is realized with graphs corresponding to an orthology relation between genes. The resulting clusters, interpreted as functional classes of proteins, permit to complement this relation.*

Keywords : Graph partitioning, Clustering by density, Orthology relation, Functional classes of proteins.

1 Introduction

With the development of the sequencing of complete bacterial genomes, we know all the gene sequences of more than a hundred species. But in many cases, their protein functions remain unknown. Establishing these functions is one of the priority tasks in genomics. The biological hypothesis is that proteins encoded by genes inherited from a common ancestral gene have identical or similar functions. Hence, the detection of these phylogenetic links, and the clustering of the related sequences, is an essential step for the identification of protein functions.

This evolutionary relation, called *homology* (Fitch, 1970), can be decomposed in two types of phylogenetic events : *paralogy* and *orthology*. They depend on the fact that the evolutionary path between *homolog* genes goes through a duplication event or not. This distinction is important since in term of function it is generally admitted that the proteins encoded by *orthologous* genes (without duplication) have conserved the same function whereas the proteins from *paralogous* genes (with duplication) have generally acquired different functions. Thus, the distinction between these two types is a fundamental step for the functional prediction process.

If these concepts are relatively easy to understand and to manipulate in theory, they are much more difficult to use in practice. Indeed, we do not have a direct access to these properties that are related to the evolution of unknown ancestors. They must be inferred from the sequence data available today. It can be achieved through out the computation of evolutionary trees. However, if this approach is well adapted to the analysis of small set of well conserved sequences, it is not tractable for massive comparison of thousands of distantly related genes.

One way to bypass this pitfall is to analyse the links between genes, orthology being estimated using an evolutive distance. All the gene sequences included in genome G_1 are compared to those of G_2 and only pairs of genes having the smallest distance values are retain; this relation is denoted BH (for best hit) in the following. It is not symmetrical and one can retain only *nearest reciprocal neighbors* making a BBH relation (for best bidirectional hit). Nevertheless, when paralogous genes are suspected, one can restrict the BBH relation eliminating any possible relation of paralogy between the two considered genes. This

*IML and LCB-CNRS, 31 Ch. J. Aiguier, 13009 Marseille (France). Mail : tristan.colombo@cmi.univ-mrs.fr

†IML-CNRS, 163 Av. de Luminy, 13009 Marseille (France). Mail : guenoche@iml.univ-mrs.fr

‡LCB-CNRS, 31 Ch. J. Aiguier, 13009 Marseille (France). Mail : quentin@ibcg.biotoul.fr

reinforced relation, called *isorthology* (Fitch, 2000), is inferred between genes A and B , respectively from genomes G_1 and G_2 , looking for the nearest genes, A' from A in G_1 and B' from B in G_2 , considered as their paralogs. If none of the two pairs of paralogs have a shorter distance than between genes A and B , that is if $d(A, A') > d(A, B)$ and $d(B, B') > d(A, B)$, then A and B are qualified *isortholog*.

These relations enable us to construct a graph Γ where the vertices are all the genes of the considered genome set. As the orthology relation is by definition a transitive one, graph Γ should consist of disjoint complete subgraphs, that are disjoint cliques. Because of errors estimating evolutionary distances, the connected components are generally not cliques and genes having different functions may be observed in the same component because of artificial edges. Consequently, the orthologous classes can be detected looking for *dense areas* in Γ , that design classes of vertices having a huge percentage of internal edges. These zones, also called *quasi-cliques* (Matsuda and al., 1999), may constitute hypothetical functional classes.

The partitioning problem in graphs has a long history we don't detail here. It becomes important with the pagination of electronic circuits (Fraissex and Kuntz, 1992), and VLSI design (Alpert and Kang, 1995). There is also many contributions linked to the graph drawing problem (Batagelj et al., 1999). It has been reactivated these last years with the analysis of large networks in several domains, like WEB (Moody, 2001), social networks (Seidman, 1983, Newman, 2001) or the treatment of biological data. In this domain, clustering is a central problem for the analysis of gene microarray data (Getz et al., 2000) or to extract functional clusters from interaction graphs between proteins. For this latter problem, most of the time, an appropriate distance is computed - generally the length of a shortest paths (Rives and Galitski, 2003) or the distance of Czekanovski-Dice (Brun et al., 2003) - and a clustering method just using this distance is applied. Sometimes, a score function is evaluated for each edges, and a peeling algorithm (Girvan et al., 2002), removing edges having maximal score, establishes clusters as connected components.

The clustering methods based on density have been introduced by Wishart in 1976; the idea is to build classes around elements having many neighbors in a threshold graph associated to a distance value. They have not been largely used, because the simple degree as density function gives unexpected results. Recently, several authors - Bader and Hogue, (2003) for simple graphs, Rougemont and Hingamp (2003) for threshold weighted graphs - have reactivated this approach, without comparison between density functions. The proposed method is also based on the evaluation of a density function for each vertex. However, it differs from the other approaches since we can only search for high density classes, even if some elements remain unassigned to a class. Our algorithm is progressive with, in a first step, the identification of kernels defined as connected vertices having a locally maximal density and, in a second step, the extension of the kernels through out the addition of connected vertices according to two strategies; the first one establishes partial classes, and the second build a complete partition. The number of classes is not required first as in many other algorithms; the number of kernels is kept as the number of classes. More, we compare the efficiency of several density functions, testing them on simple graphs with a given number of dense zones that are recovered.

The choice of a density function is critical for the efficiency of the algorithm. So, we will first describe in section 2 four density functions. The algorithm is detailed in section 3. The performances of the different functions are compared by simulations on random graphs in section 4. In section 5, an application to a set of real sequence data illustrates the relevance of this method in a biological context.

2 Density functions

Let X be the set of the n vertices, E the set of the m edges and $\Gamma = (X, E)$ the corresponding graph. It may be assumed to be connected; otherwise we treat separately its different connected components. For any part Y of X , let $\Gamma(Y)$ be the set of vertices out of Y that are adjacent to Y

$$\Gamma(Y) = \{x \in X - Y \text{ such that } \exists y \in Y, (x, y) \in E\}.$$

The neighborhood of x is $\Gamma(x)$. The degree of a vertex x is denoted $Dg(x) = |\Gamma(x)|$ and let δ be the maximum degree in the graph.

For each vertex x , we evaluate a density value denoted $De(x)$. The density function De is a map from X to \mathbb{R}_+ varying increasingly with the number (or the percentage) of edges in the neighborhood of a vertex. By definition, all the vertices having degree 1 will get a density equal to 0, to avoid inappropriate or undefined values in the following computations. We propose four functions, which will be compared in section 4 :

- The historical one was the degree ; here, we normalize it :

$$De_1(x) = \frac{Dg(x)}{\delta}.$$

This function gives a central place to vertices with a high degree, a place they don't necessarily have, especially in protein graphs.

- The average degree in the neighborhood of x :

$$De_2(x) = \frac{Dg(x) + \sum_{y \in \Gamma(x)} Dg(y)}{(1 + Dg(x))}.$$

This function counts the same way the edges adjacent to x and those that are adjacent to a neighbor of x . In order to overcome this drawback, we consider :

- The rate of triangles going through x . Let $N_t(x)$ be the number of triangles where x is a vertex :

$$N_t(x) = |\{(y, z) \in E \text{ such as } (x, y) \in E \text{ and } (x, z) \in E\}|.$$

This number is divided by the maximum value expected for a vertex of degree $Dg(x)$.

$$De_3(x) = \frac{N_t(x)}{\frac{1}{2} \cdot Dg(x) \cdot (Dg(x) - 1)}.$$

This function is the most often used in similar approaches (Bader and Hogue, 2003). A vertex having all its neighbors connected, making so a clique, will have a maximum density value equal to 1.0. At the contrary, when some vertex pairs in $\Gamma(x)$ are not connected, this function decreases very quickly. In order to give more density to the vertices which have many links, we introduce a new function :

- The percentage of edges in the neighborhood of x , that is the number of edges adjacent to x plus those making triangle, divided by the maximal number of edges in the neighborhood of a vertex of degree $Dg(x)$.

$$De_4(x) = \frac{Dg(x) + N_t(x)}{\frac{1}{2} \cdot Dg(x) \cdot (Dg(x) + 1)}.$$

Another density function has been tested ; first the Czekanovski-Dice distance on Γ is evaluated and the density is estimated from the average distances in the neighborhood of any vertex. It provides also satisfying results which are not detailed here, because they are not better than those of De_4 .

In order to evaluate the first density function we check the edge list whose length is bounded by $m \leq n\delta$ but, for the other ones, we must test the edges in the neighborhood of x which contains at most δ vertices. The computation of the density function is thus in $O(n\delta^2)$.

3 Algorithm for dense classes

The searched classes must be connected in Γ , their elements sharing high density values. Our initial idea was to select a density threshold and to consider the partial subgraph whose vertices have a density greater than this threshold ; thus the classes would be the connected components of this threshold graph. This method does not give the expected results for two reasons :

- The choice of a threshold is a critical problem ; when it is equal to the maximum density value, if it is unique, there is only one class with one element. A threshold equal to 0, gives only one class containing all the elements. An intermediate value, for instance the average density in the graph, does not always give the number of expected classes (cf section 4).
 - Enumerating all the possible threshold values, we have observed that often none was satisfying, in particular for functions De_1 and De_2 . By decreasing the threshold, we often obtain only a single growing class, and many singletons. For functions De_3 and De_4 there is a lot of fluctuations for the number of classes corresponding to the high values, but they generally contain very few elements.
- Since there is no straightforward way to fix a threshold, we consider the *local maximum values* of the density function.

3.1 Kernels of the classes

A kernel, denoted K , is a connected part of Γ , obtained by the following algorithm : we start from the local maximum values of the density function that are larger than the average and we consider the partial subgraph of Γ reduced to those vertices.

$$\forall x \in K, \forall y \in \Gamma(x) \text{ we have } De(x) \geq De(y).$$

The initial kernels are the connected components of this graph. More precisely, if several vertices with maximum value are adjacent, they necessary have the same density value and they belong to the same kernel ; otherwise the initial kernels are singletons. We will keep the number of classes as the number of initial kernels. Then, we assign recursively to each kernel K all the vertices (i) having a density greater than or equal to the average density value over X and (ii) adjacent to only one kernel. Doing so, we avoid any ambiguity in the assignment, postponing the decision when several are possible.

3.2 Extensions to dense classes

In a second step, we extend the kernels by adding other elements, those that could be assigned to several ones or those that have a density lower than the average. We have implemented the two following strategies :

3.2.1 Partial extension

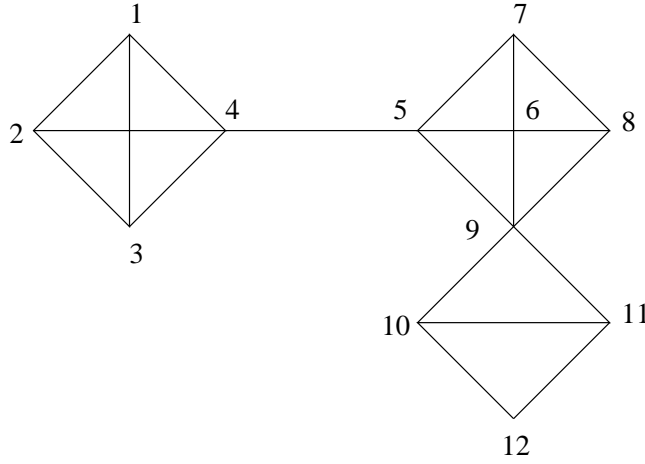
The principle of this extension rule is to aggregate to a kernel all the connected elements while its average degree increases. Let C be a class initially restricted to a kernel K ; we compute :

- the average degree of C , considering only the internal edges, $\overline{Dg}(C) = \frac{2}{|C|}|\{(x, y) \in C \times C\}|$
- then for each element x of $\Gamma(C)$, the number $c_x = |\Gamma(x) \cap C|$,
- the maximum c_{max} of the c_x values over $\Gamma(C)$.

If $c_{max} \geq \overline{Dg}(C)$, we add to the class all the elements having c_{max} connections. If at least one element is added, this procedure is repeated until there is no element increasing the average inner degree.

Thus, the classes are still connected parts. The density values have been used to initialize each class which grows step by step. Generally, at the end of this procedure all the vertices are not clustered, but these partial classes would have a density value higher than those making a complete partition. We also observe that an element may be added to several kernels and so the final classes are not necessarily disjoint.

Example : The kernels in the graph of Figure 1 have been computed using the density function De_4 . The local maximal values correspond to vertex sets $\{1, 2, 3\}$, $\{7, 8\}$ and $\{12\}$ and the average over X is 0.83. So only vertices 10 and 11 may be added to the third kernel ; they are printed below before the first sign +. For the extended classes, vertex 4 is in the first class ; for class 2, vertex 6 enables to increase the average degree then vertices 5 and 9, which have two links with $\{6, 7, 8\}$, are also added. For class 3, vertex 9 is also linked. After the arrow, we indicate between parentheses the internal average degree of the resulting class :



X	1	2	3	4	5	6	7	8	9	10	11	12
Dg	3	3	3	4	4	4	3	3	5	3	3	2
De_4	1.0	1.0	1.0	0.7	0.6	0.8	0.833	0.833	0.533	0.833	0.833	1.0

FIG. 1 – Graph with $n = 12$ and $m = 20$; the table indicates the degree and the density value of each vertex

Class 1 : 1 2 3 + 4 \rightarrow (3.0)

Class 2 : 7 8 + 6 5 9 \rightarrow (3.2)

Class 3 : 10 11 12 + 9 \rightarrow (2.5)

We note that vertex 9 has been added twice according to Figure 1, where it is an articulation point.

3.2.2 Complete extension

As we shall see, this first strategy tends to assign a quarter of the elements in the kernels and half the elements in the extended classes (cf section 4). In order to cluster all the vertices, we have developed another strategy for the kernel extension.

Let us consider that we have p kernels denoted K_i with s_i elements. Let $L = X - \bigcup_{1 \leq i \leq p} K_i$ be the set of the q vertices that remain to be classified. We sequentially assign them to the kernels to which they are mainly connected, whatever their density value is.

The elements in L are examined in the decreasing density order. Let x be the current vertex :

– for each kernel K_i we compute the number $c_i(x)$ of its connections to x :

$$c_i(x) = |\Gamma(x) \cap K_i|$$

– x is connected to the kernel K_j so that $c_j(x)$ is maximal and, in case of several maximum, to the one for which s_j is minimal;

– the quantities c_i and s_i are updated.

Doing so, each element is assigned to a single kernel. The decision in case of ties to place x in the class with the smallest number of internal edges, will increase the number of elements in small classes. This rule tends to give balanced classes.

Example : Considering again the graph in Figure 1 and its kernels $\{1, 2, 3\}$, $\{7, 8\}$ and $\{10, 11, 12\}$, we obtain :

$L = (6, 5, 4, 9)$

Class 1 : 1 2 3 + 4
 Class 2 : 7 8 + 6 5 9
 Class 3 : 10 11 12 +

Note that vertex 9 has three connections with class 2 and two with class 3. So, it is assigned to class 2.

3.3 Complexity

Kernel computation is in $O(m) \approx O(n\delta)$. For the extension step we start by evaluating the average internal degree of each kernel and the number of connections of the adjacent elements; this part is in $O((n - q)\delta)$.

- In the first case, at each iteration, we retain the elements having a sufficient degree and we update the average class degree; this procedure has complexity $O(pq\delta)$. The number of iterations being bounded by δ , the time complexity of the first extension rule is in $O(pq\delta^2)$;
- in the second case, we affect at each iteration only one element, and we update the p values c_i and s_i examining at most δ edges, which gives $O(pq\delta)$ for all the classes.

Noting that $q < n$ and that the kernel step occurs only once, the complexity of the extension procedure is in $O(np\delta^2)$.

Compared to other methods computing first distance values between vertices or a score function, as the number of shortest paths going through any edge (Newman, 2001), our algorithm is very efficient. It allows the treatment of large graphs, which is essential in biological context where data accumulate at exponential growth rates; in one year, the number of entirely sequenced genomes go from sixty to more than a hundred, each one possessing several thousands of genes; the linearity in n becomes essential.

4 Validation by simulation

In order to evaluate the ability of the method to recover high density areas in a graph, we compare the performances of the four density functions and the two extension procedures. First, we have developed a random generator of graphs in which there are classes having more internal edges than the external ones. Then we apply our algorithm for different values of the following parameters.

4.1 Generator of random graphs

The generator of random graphs depends four parameters :

- N : the number of vertices,
- p : the number of initial classes in the graph,
- d_i : the average internal density within the classes,
- d_e : the average density of the external edges.

In order to get such a random graph, we begin with a random partition of the N elements in p classes, denoted C_1, \dots, C_p . This initial partition P is stored in a vector p_1, \dots, p_N , where p_k is the class number of the k -th element. Next, for each pair of elements, we select at random a real number between 0 and 1, and we add the corresponding edge if and only if this number is lower than or equal to d_e (resp. d_i) when the two elements are in different (resp. identical) classes. This procedure does not guarantee that our graphs will have precisely p dense classes; they may be decomposed, or rearranged according to the random edges. Similarly, the real density values are not necessarily equal to d_i and d_e but we have observed that, on average, these parameters are correctly fitted.

4.2 Quality of the classes compared to the initial partition

They are three levels of classes successively built :

- the initial kernels,
- the classes corresponding to the extended kernels (we have suppressed the multiple assignment possibility which is very rare)
- the classes obtained by full assignment to the most connected kernel.

Let N_c be the number of classified vertices at each level. They are distributed in p' classes denoted $C'_1, \dots, C'_{p'}$, realizing a partition P' . We first aim at mapping the classes of P' onto those of P evaluating $n_{i,j} = |C_i \cap C'_j|$. We set that the *corresponding* class to C'_j is the class C_k in P , that contains the greatest number of elements of C'_j , that is the one such that $n_{k,j} \geq n_{i,j}$ for all i from 1 to p .

In order to evaluate the accuracy of the computed classes, we evaluate three criteria.

- τ_c : the percentage of clustered elements in P' ($\frac{N_c}{N}$).
- τ_e : the percentage of elements in one of the p' class which belong to its corresponding class in P .
- τ_p : the percentage of pairs of clustered elements belonging to one class in P' which are also together in P .

The first criterion estimates the efficiency of the clustering process at each level; if very few elements are clustered, the method will be inefficient. For the second criterion, we must compute, for each class in P' , the distribution of the elements of any initial class to define its corresponding class in P . Thus it can be interpreted as a percentage of "well classified" elements. For the third one, we compare the initial numbers of joined pairs in P' that come from a single class in P . So, high score values are expected for these three criteria.

Remark : The two last criteria may reach their maximum value (1.0) even when the partitions P and P' are not identical, for instance when two classes of P' are included in one of P . If several classes of P' have the same corresponding class in P all their elements will be considered as well classified; and the rate of pairs will also be equal to 1.

4.3 Results

These results are average values obtained from 200 graphs of 100 vertices distributed in 3 classes, with an internal density $d_i = 0.5$ and an external density $d_e = 0.1$. Such a gap seems to give easy problems, that are classes easy to recover. But assuming that the p classes have the same cardinality, there are $d_i \frac{n(n-p)}{2p}$ intra-class edges and $d_e \frac{n^2(p-1)}{2p}$ inter-class edges. So there will be around $808 + 333$ edges in our graphs, corresponding to an average density of 0.232. It means that the internal density is approximatively twice the average over the whole graph.

The rows of Table 1 correspond to the three types of computed classes and the columns to the four density functions. Each cell contains the values of the three criteria τ_c , τ_e and τ_p . The last row indicates the average number of classes in P' .

	De_1			De_2			De_3			De_4		
	τ_c	τ_e	τ_p	τ_c	τ_e	τ_p	τ_c	τ_e	τ_p	τ_c	τ_e	τ_p
Kernels	.26	.78	.65	.21	.89	.80	.26	.96	.93	.25	.95	.92
Partial extension	.42	.84	.74	.38	.91	.85	.48	.97	.95	.50	.97	.95
Complete extension	1.0	.68	.59	1.0	.65	.55	1.0	.93	.91	1.0	.95	.93
Nb. of classes	2.4			3.5			4.6			5.4		

Table 1 - Average results obtained from 200 graphs of 100 vertices distributed in three classes. They are randomly generated with an internal density $d_i = 0.5$ and an external density $d_e = 0.1$

The superiority of functions De_3 and De_4 is obvious. The function De_1 is not enough discriminant to generate a sufficient number of local maximum values. Since they determine the number of kernels, the classes are badly defined compared with the initial ones. The performances of functions De_3 and De_4 are satisfying, at each level. On average, 25% of the elements belong to the kernels and 50% are the extended classes. More than 95% of the joined elements are in the same initial class. For the complete extension, more than 90% of the assignments are correct.

4.4 Extended results

In order to study the variation of the results on more difficult problems we fix the external density (equal to .1 then .2) and the internal one varies from .4 to .7 (an internal density greater than .7 gives easy problems). The average results of function De_4 are established again on 200 graphs of 100 vertices (cf Table 2).

When there are few connections between classes ($d_e=.1$) whatever is the internal density, the results are satisfying : 25% to 31% of the elements are classified in kernels and 44% to 59% in the extended classes. Globally, more than 90% of the assignments are correct. But when the number of external links increases ($d_e=.2$), the results are not so good, except when the internal density is high. When the gap between the density values is lower than .3, one can think that the classes are not clearly established.

d_e	d_i	Kernels			Partial ext.			Complete ext.		
		τ_c	τ_e	τ_p	τ_c	τ_e	τ_p	τ_c	τ_e	τ_p
.1	.4	.25	.93	.88	.44	.94	.90	1.0	.89	.84
.1	.5	.27	.96	.93	.48	.97	.95	1.0	.94	.91
.1	.6	.28	.97	.96	.52	.98	.97	1.0	.95	.93
.1	.7	.31	.98	.97	.59	.99	.98	1.0	.95	.93
.2	.4	.24	.74	.57	.42	.74	.60	1.0	.62	.49
.2	.5	.25	.85	.75	.44	.88	.80	1.0	.76	.66
.2	.6	.25	.91	.85	.45	.93	.89	1.0	.80	.72
.2	.7	.27	.93	.90	.50	.96	.93	1.0	.81	.74

Table 2 - Average results for function De_4 obtained from 200 graphs of 100 vertices distributed in 3 classes when the internal density varies.

We have also tested our algorithm when increasing the number of vertices to $N = 300$ and the number of classes to 5, comparing the density functions De_3 and De_4 with $d_i = 0.5$ and $d_e = 0.1$. We always obtain approximately the same results : more than 20% of the elements are clustered in kernels and more than 43% of the elements are classified in the extended classes with more than 90% of correct assignments.

On the contrary, considering many small classes (for instance $N = 200$ and 15 classes), the percentage of classified elements stay approximately the same, but only 63% of the assignments are correct. Indeed, when the number of classes increases while the vertex set remains the same, the class structure is much more difficult to retrieve.

5 Application to the ABC transporters

The members of the periplasmic binding proteins extracted from the ABCDB database (Quentin and Fichant, 2000) have been used to illustrate the performances of the method. These proteins are partners of ABC transporters – transporters mediating the uptake of nutrients by bacteria. Each bacterial genome include family of paralogous genes encoding for multiple systems importing different molecules. These genes fall into a small number of families according to protein sequences similarities. From the analysis of the available experimental data, it appears that members of each family are involved in the selection of related molecules (i.e. amino acids, sugars, ions...). However, this partition must be refined in order to obtain more precise functional predictions (i.e. the amino acid is the alanine, the sugar is the maltose and the ion is manganese). Here, we will use as an illustration one of the subfamily previously reported (Quentin et al., 1999). This family referred as to S_5 is among the largest ones in bacteria. It appears heterogeneous and, based on the analysis of a set of 20 annotated genomes (Quentin et al., 2002), it has been further subdivided in five sub-families according to protein sequence similarities (S_{5a} , S_{5b} , S_{5c} , S_{5d} , and S_{5f}). The solute binding proteins of the recently sequenced genomes have been annotated according to this classification.

	S_5	S_5a	S_5b	S_5c	S_5d	S_5f	
1	2	–	–	–	–	–	2
2	–	–	56	–	–	–	56
3	16	–	–	14	–	–	30
4	8	19	–	–	–	–	27
5	9	100	–	–	–	–	109
6	10	10	–	–	–	–	20
7	17	84	–	–	–	–	101
8	17	–	–	78	–	–	95
9	4	32	–	–	–	–	36
10	50	–	–	–	–	–	50
11	4	–	–	–	–	–	4
12	13	–	–	9	–	–	22
13	4	–	–	–	112	–	116
14	–	–	–	–	–	45	45
15	5	–	–	–	–	–	5
16	4	–	–	3	–	–	7
17	19	–	–	6	–	–	25
18	5	–	–	–	10	–	15
	164	245	44	110	122	45	765

TAB. 1 – Distribution of the SBP proteins of the six sub-families of the family 5 of ABC transporters in the 18 computed classes.

Here, we used this enlarged sample of 765 sequences extracted from 95 genomes in order to test the ability to recover of the sub-families. We have used three different similarity relations between genes : the best score (BH), the best reciprocal score (BBH), and the isorthology relations. The choice of the similarity relation depends on the nature of the problem to solve and allows different levels of analysis. Ideally, the isorthology should make it possible to define groups of ABC transporters sharing the same specificity of transported molecules, the *substrate*, whereas with BH and BBH relations, the ABC transporters should have similar substrates.

The number of connected components (the minimal number of classes) depends on the selected relation. With the BH relation we obtain a single one connected component, eleven connected components with the BBH relation, and forty connected components with the isorthology relation (about half of the connected components with BBH and isorthology relations contain less than five elements). Classifications remain compatible : their class intersections are almost perfect. We present in details the results obtained with the BH relation on family 5 because they can be directly confronted to the published classifications. According to the simulations, we use the De_4 function. Figure 2 shows the corresponding graph drawn by the Pajek software (Batagelj, 2001) and the results are summarized in table 1 : in column, the sub-families described in Quentin and Fichant, 2000 (the first column, named S_5 includes elements recognized as S_5 members, but that could not further be attributed to one of the five sub-families) ; in row, the 18 classes established by our algorithm. We note that unclassified elements from the S_5 sub-family are now affected to our classes. The sub-families S_5b , S_5f are confirmed, and the sub-family S_5d is partially confirmed : only 10 genes are assigned to another sub-family. The sub-families S_5a and S_5c have been broken up into five classes (respectively 4, 5, 6, 7, 9 – and 3, 8, 12, 16, 17). When experimental data are available, a specific substrate can be assigned to classes (i.e. spermidine/putrescine for class 4 and sulfate for class 9).

In conclusion, the results obtained with our method are consistent with the previous classification proposed by experts. We are able to specify the classification of 164 elements and we subdivide the larger sub-families in smaller classes that received experimental supports.

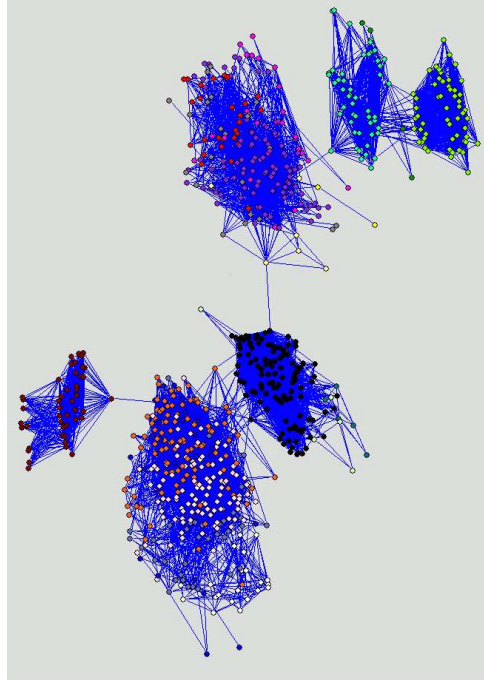


FIG. 2 – Graph of the SBP of the family 5. The genes are linked by a BH relation. The different colors of vertices indicate the classes obtained using the De_4 function and the complete extension of classes.

Acknowledgements

This work has been realized with the help of the French program inter-EPST "BioInformatique" followed by the ACI-IMPBio. We also thank an anonymous referee for its meticulous reviewing.

6 Bibliography

- C.J. ALPERT AND A. KANG (1995) Recent direction in netlist partitioning : a survey, *Integration : the VLSI Journal*, 19, 1-2, 1-81.
- G.D. BADER AND C.W. HOGUE (2003) An automated method for finding molecular complexes in large protein interaction networks, *BMC Bioinformatics*, 4, 2, 27 p.
- V. BATAGELJ AND M. MRVAR (1999) Partitioning approach to visualisation of large graphs, *Lecture Notes in Computer Science* 1731, Springer, 90-97.
- V. BATAGELJ (2001) Pajek – program for large networks analysis and visualization, In *Link Analysis and Visualization*.
- C. BRUN, A. GUÉNOCHE, B. JACQ (2003) Approach of the functional evolution of duplicated genes in *Saccharomyces cerevisiae* using a new classification method based on protein/protein interaction data, *J. of Structural and Functional Genomics*, 3, 213-224.
- M.B. EISEN, P.T. SPELLMAN, P.O. BROWN, D. BOTSTEIN (1998) Cluster Analysis and display of genome wide expression patterns, *Proc. Natl. Acad. Sci. USA*, 95, 14863-14868.
- W. M. FITCH (1970) Distinguishing homologous from analogous proteins, *Syst. Zool.*, 19, 99-113.
- W. M. FITCH (2000) Homology : a personal view on some of the problems, *Trends in Genetics*, 16.
- H. DE FRAISSEX AND P. KUNTZ (1992) Pagination of large scale networks ; embedding a graph in R_n for effective partitioning, *Algorithmic review*, 2(3), 105-112.
- G. GETZ, E. LEVINE, E. DOMANY (2000) Coupled two-way clustering analysis of gene microarray data, *Proc. Natl. Acad. Sci. USA*, 97, 12079-12084.

- M. GIRVAN AND M.E.J. NEWMAN (2002), Community structure in social and biological networks, *Proc. Natl. Acad. Sci. USA*, 99, 7821-7826.
- C. F. HIGGINS (1992) ABC transporters : from micro organism to man, *Annu. Rev. Cell Biol.*, 8, 67-113.
- Y. QUENTIN AND G. FICHANT (1999) Inventory, assembly and analysis of *Bacillus subtilis* ABC transport systems, *J. Mol. Biol.*, 287, 467-484.
- Y. QUENTIN AND G. FICHANT (2000) ABCDB : an ABC transporter database, assembly and analysis of ABC transport systems in complete genomes, *J. Mol. Microbiol. Biotechnol.*, 2, 501-504.
- Y. QUENTIN, J. CHABALIER AND G. FICHANT (2002) Strategies for the identification, the assembly and the classification of integrated biological systems in completely sequenced genomes, *Computer and Chemistry*, 26, 447-457.
- H. MATSUDA, T. ISHIHARA, A. HASHIMOTO (1999) Classifying molecular sequences using a linkage graph with their pairwise similarities, *Theoretical Computer Science*, 210, 305-325.
- J. MOODY (2001) Identifying dense clusters in large networks, *Social Networks*, 23, pp. 261-283.
- M.E.J. NEWMAN (2001), Scientific Collaboration Networks : Shortest paths, weighted networks and centrality, *Phys. Rev.*, 64.
- A.W. RIVES AND T. GALITSKI (2003) Modular organization of cellular networks, *P.N.A.S.*, 100, 1128-1133.
- J. ROUGEMONT AND P. HINGAMP (2003) DNA microarray data and Contextual analysis of correlation graphs, *submitted*.
- S.B. SEIDMAN (1983) Network structure and minimum degree, *Social networks*, 5, 269-287.
- D. WISHART (1976) Mode analysis : generalization of nearest neighbor which reduces chaining effects, *Numerical taxonomy*, Academic Press, 282-311.