UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ALEXANDER PINARES BOLIVAR

A Bitwise Clique Detection Approach for Accelerating Power Graph Computation and Clustering Dense Graphs

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. João Luiz Dihl Comba

Porto Alegre
2020

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

, Alexander Pinares Bolivar

A Bitwise Clique Detection Approach for Accelerating Power Graph Computation and Clustering Dense Graphs / Alexander Pinares Bolivar . – Porto Alegre: PPGC da UFRGS, 2020.

56 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Orientador: João Luiz Dihl Comba .

1. Análises dos grafos de potência. 2. Redução de arestas. 3. Detecção de bicliques. 4. Clustering de dados. 5. Operações binarias. I. , João Luiz Dihl Comba. II. Título.

“If I have seen farther than others,
it is because I stood on the shoulders of giants.”
— Sir Isaac Newton

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor PhD. Joao Luiz Dihl Comba of the Computer Science department at Federal University of Rio Grande do Sul. The door to Prof. Comba's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

RESUMO

Os grafos são essenciais para muitas representações de dados. A análise visual de grafos é usualmente difícil devido ao tamanho, o que representa um desafio para sua visualização. Além de isso, seus algoritmos fundamentais são frequentemente classificados como NP-difícil. Análises dos grafos de potência (PGA em inglês) é um método que simplifica redes usando representações reduzidas para subgrafos completos chamados cliques e subgrafos bipartidos chamados bicliques, em ambos casos com una redução de arestas. Os benefícios da representação de grafo de potência são a preservação de informação e a capacidade de mostrar a informação essencial sobre a rede original. Entretanto, encontrar uma representação ótima (a máxima redução de arestas possível) é também um problema NP-difícil. Neste trabalho, propomos BCD, um algoritmo guloso que usa um abordagem de detecção de bicliques baseado em operações binarias para encontrar representações de grafos de potencia. O BCD é mas rápido que as estratégias atuais da literatura. Finalmente, descrevemos como a estrutura induzida pelo grafo de potência é utilizado para as análises dos grafos densos na detecção de agrupamentos de nodos.

Palavras-chave: Análises dos grafos de potência. redução de arestas. detecção de bicliques. clustering de dados. operações binarias.

A Bitwise Clique Detection Approach for Accelerating Power Graph Computation and Clustering Dense Graphs

ABSTRACT

Graphs are at the essence of many data representations. The visual analytics over graphs is usually difficult due to their size, which makes their visual display challenging, and their fundamental algorithms, which are often classified as NP-hard problems. The Power Graph Analysis (PGA) is a method that simplifies networks using reduced representations for complete subgraphs (cliques) and complete bipartite subgraphs (bicliques), in both cases with edge reductions. The benefits of a power graph are the preservation of information and its capacity to show essential information about the original network. However, finding an optimal representation (maximum edges reduction) is also an NP-hard problem. In this work, we propose BCD, a greedy algorithm that uses a Bitwise Clique Detection approach to finding power graphs. BCD is faster than competing strategies and allows the analysis of bigger graphs. For the display of larger power graphs, we propose an orthogonal layout to prevent overlapping of edges and vertices. Finally, we describe how the structure induced by the power graph is used for clustering analysis of dense graphs. We demonstrate with several datasets the results obtained by our proposal and compare against competing strategies.

Keywords: Power Graph Analysis, edges reduction, biclique detection, clustering analysis, binary operation.

LISTA DE ABREVIATURAS E SIGLAS

PGA	Power Graph Analysis
BCD	Biclique and Clique detection
FSBAS	Finding the similarity between adjacency sets.
BFS	Breadth-first search
DFS	Depth First Search
BMP	Windows bitmap
PNG	Portable Network Graphics
JPG	Joint Photographic Group

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUCTION

Through the years, scientists have developed several methods to study real-world networks. In neuroscience, for example, there is great interest in finding information on the behavior of neural networks (SPORNS; TONONI; KÖTTER, 2005; SPORNS, 2011; REUS; HEUVEL, 2013). A prominent feature in this field is the study of connectomes, but the complexity of living things makes it impossible to study individual neurons in most cases. To reduce the complexity, the analysis is done at specific brain regions, like the human connectome (SPORNS; TONONI; KÖTTER, 2005) or the cat connectome (REUS; HEUVEL, 2013). Other networks widely studied are the protein (CHASSEY et al., 2008), social (BRANDES; WAGNER, 2004) and citations networks (LESKOVEC; HORVITZ, 2014).

The Power Graph Analysis (PGA) is a technique proposed to simplify graphs using reduced representations, called *Power Graphs*, that has special representations for complete subgraphs (cliques) and complete bipartite subgraphs (bicliques). Power graphs allow a considerable reduction in the number of edges. In Figure 1.1 we illustrate a dense graph (a) and its corresponding power graph representation (b), which uses the special terminology of *power nodes* and *power edges* to refer to its vertices and edges. There are many possible power graphs associated to a given graph, and an optimum power graph is the one that has the smallest number of power edges, which is directly related to the success on finding bicliques and cliques. This problem is likely to be NP-hard since finding the maximum biclique on a graph was proved to be NP-hard (PEETERS, 2003). Therefore, the power graph computation algorithms rely on approximation algorithms and heuristics.

A hierarchical clustering approach is used in (ROYER et al., 2008) for power graph computation, but it is slow for dense graphs. Dwyer et al. (DWYER et al., 2014) proposed an improved algorithm using a greedy strategy and which allowed power graphs to be used for the visualization of dense graphs. However, their results were limited to small graphs (up to 100 vertices).

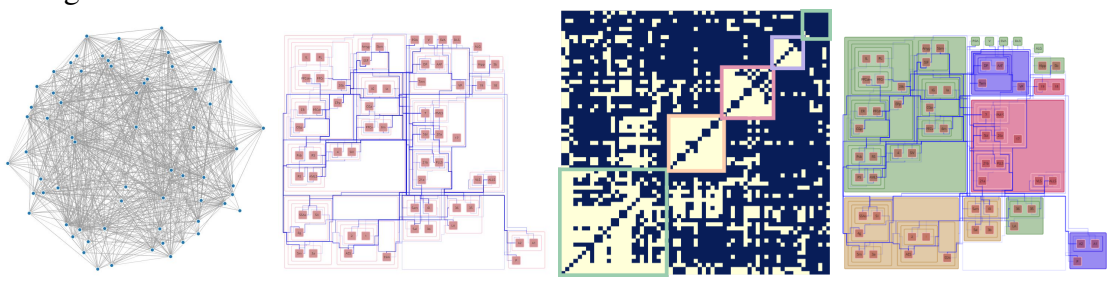
In this dissertation, we address the problem of scaling the analysis of power graphs to larger graphs. Our approach starts with the proposal of a bitwise greedy algorithm to find power graphs with reduced number of edges. Our algorithm runtime is approximately one order of magnitude faster than the improved method by (DWYER et al., 2014). The larger power graphs also pose difficulties with respect to its visualization. We designed

a customized algorithm for building orthogonal representations of power graphs. Finally, we describe how the power graph can be used to drive a clustering algorithm defined over a graph. We demonstrated that clustering results can be better understood when using the orthogonal layout of the power graph, which allows to evaluate the relation among elements in a given cluster.

In summary, the main contributions introduced in this work are:

- Binary Clique Detection (BCD), a PGA method that uses a bitwise matrix representation to improve clique and biclique detection, and lead to a speedup of an order of magnitude to previous methods,
- An orthogonal layout specially designed for power graphs, and
- A power-graph algorithm for clustering dense graphs that combined with the orthogonal layouts allow inspecting clustering results, as well as the connectivity among nodes in each cluster.

Figura 1.1: Clustering process based on power graph analysis applied to the graph representing the cat connectome.



(a) Dense Graph (b) Orthogonal Layout of Power Graph (c) Power Graph Adjacency matrix (d) Clusters found with BCD

1.1 Motivation

The topological information about problems in graph theory (NP-complete and NP-hard) are used to the prediction, classification and visualization systems, but their runtimes for real-world networks is extremely high. In the last years, several heuristics are developed for finding near-optimal solutions, their advantages are the reduction of results and the runtime, making the analysis of real-world networks practical. In this sense, we can obtain relevant information about real network represented by graphs and note that countless works are published in Information visualization, there are few about graph simplification, especially for dense graphs.

1.2 Structure of this document

The remaining of this thesis is organised as follow: Chapter 2 presents the related works, we explain concepts about power graphs in Chapter 3; data structure, their description and evaluation are presented in Chapter 4; Chapter 5 describes the power graph, orthogonal layout, clustering and image compression algorithms exhibiting their results in Chapter 6; Chapter 7 and Chapter 8 describe conclusions and future works respectively.

2 POWER GRAPH

In this chapter, we present the background for power graph and related work.

2.1 Background

In this section, we review the terminology associated to power graphs: graph, clique, biclique, star and dendrogram. Also the analysis of the structured data and finally a real example for the protein-protein interaction of Saccharomyces cerevisiae.

2.1.1 Graphs, Cliques, Bicliques and Stars

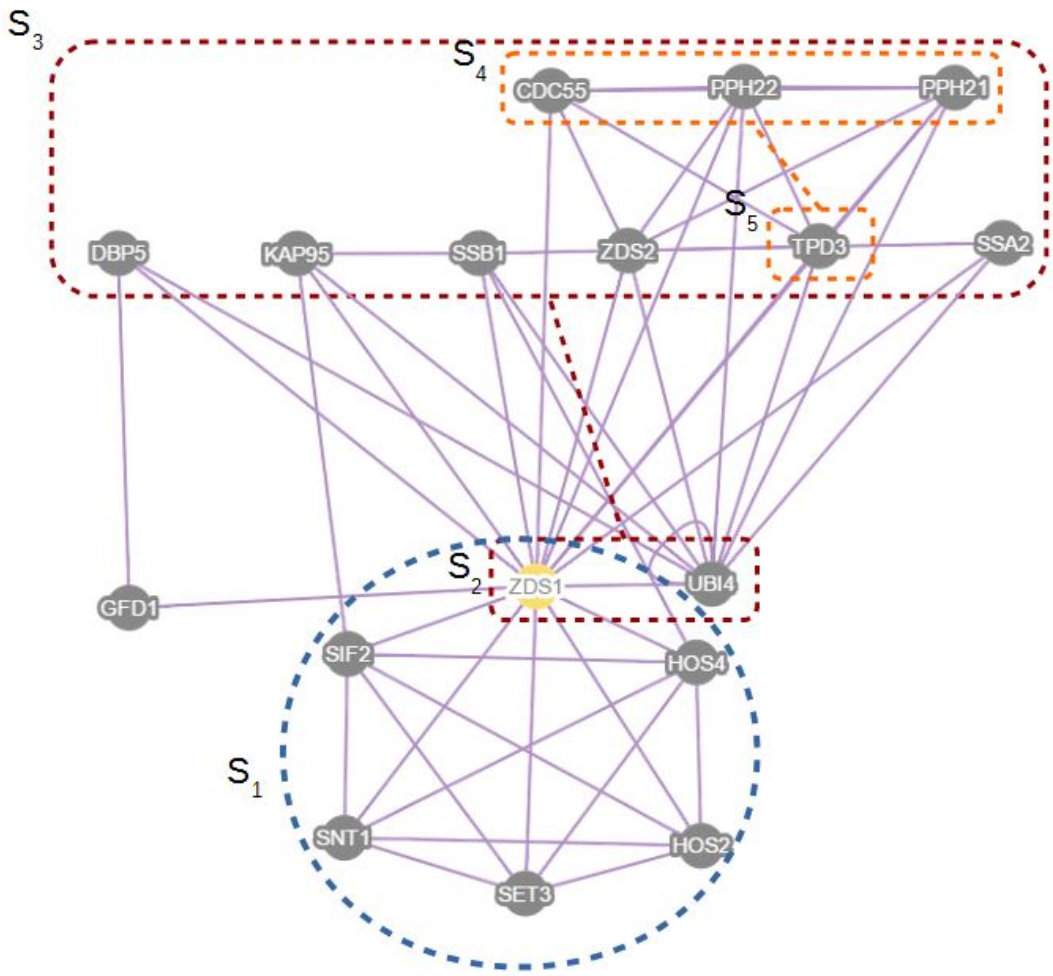
Let a graph $G = (V, E)$ be an ordered pair of a non-empty finite set $V = V(G)$ of vertices and a finite set of edges $E = E(G)$ defined as unordered pairs of distinct vertices (BONDY; MURTY, 1976). A clique is defined as a subset of vertices of an undirected graph such that the generated subgraph is complete. A graph is bipartite if its vertices can be divided into two non-empty disjoint sets, such that their edges connect the first to the second set. A complete bipartite graph, or biclique, is a bipartite graph where every vertex of the first set is connected to every vertex of the second set. A star is a complete bipartite graph $G_S = (V_1, V_2, E)$ where $|V_1| = 1$ or $|V_2| = 1$. The density of a graph is the ratio between the number of edges and the number of edges of the complete graph with the same vertices, defined as $D = (2 * |E|)/(|V| * (|V| - 1))$.

Real-world networks are attractive to analyse, as protein-protein interaction (e.g. ZDS1 Protein of Saccharomyces cerevisiae in Figure 2.1). S_1 is a clique of 6 nodes and 15 edges. Also, the biclique of S_2 and S_3 have 18 edges. Finally, the star of S_4 and S_5 have 3 edges. The graphical representation for this simple graph has many overlapped edges.

2.1.2 Power Graphs

The basic idea for the power graph representation is illustrated in Figure 2.2. The compact representation used for power graphs replaces the structures of bicliques, cliques, and stars in a given graph by a compact representation with fewer edges. The power graph

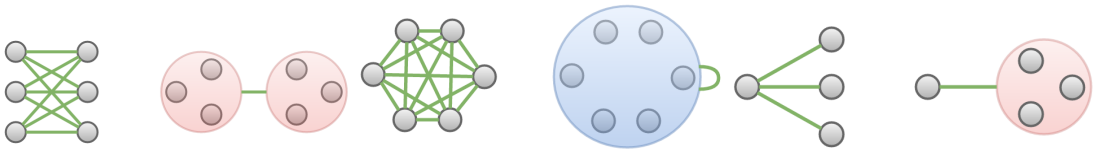
Figura 2.1: Example of a simple protein-protein interaction of Saccharomyces cerevisiae <http://www.yeastgenome.org/>, sets correspond to a clique, biclique and start.



is composed of four main elements: power nodes, power edges, and vertices and edges from the original graph. Formally, a power graph of a undirected graph $G = (V, E)$ is $PG = (PV, PE, V, E)$, where PV is a power node set, PE is a power edge set. A vertex $v \in PV \wedge$ and an edge $e \in PE$ if and only if these conditions are satisfied: v is a subset of V, $|v| > 1$, e is a pair $(u, v) \wedge u \in PV$ and $v \in \{PV \cup V\}$(a power edge can link to a power node from a vertice). In a power graph it must be true that $u \cap v \in \emptyset, u, v$ where $u, v \in PV$.

A dendrogram is a tree data representation with three types of nodes: leaf node, inner node, and a root node (PHIPPS, 1971). It is used in hierarchical clustering and in our case to store the power graph. We use the dendrogram data structure as follows: leaf nodes represent vertices of the original graph, inner nodes represent a node subset and the root node represent all the graph nodes.

Figura 2.2: Power Graph for bicliques, cliques and stars.



(a) Biclique (b) Clique (c) Star

2.1.3 Data structure

Operations between sets of vertices are essential to implement the construction of a power graph. Common power graph algorithms are required to find the number of common nodes between their adjacency sets for every pair of vertices. We refer to node similarity when two nodes share a high number of adjacent vertices. Given an adjacency list as a graph representation, the complexity of finding the similarity between adjacency sets (FSBAS) is $O(V * E)$. As an example, consider a graph of 1000 vertices with an edge density of 60%. The number of iterations for FSBAS are 300 millions approximately. Considering an adjacency matrix representation, the complexity for FSBAS is $O(V^3)$.

We use of binary adjacency matrix to store the graph, and a row of the matrix can be implemented in 4 ways for C++ language programming:

- As a boolean array implemented by *bool VariableName[N]*, where *N* is the array length.
- As an array of 64-bit integers implemented by and *unsigned long long int VariableName[N]*, in this way, a single integer can store 64 elements in a row, and for a graph with 1000 nodes, we need 16 integers to store a row of the adjacency matrix.
- As a bit-set implemented by "*bitset*<N>"where N is a constant integer.
- As a dynamic bitset implemented by "*boost::dynamic_bitset*<>".

The first and second also can be implemented using the dynamic structure called *vector*, The third was implemented in Standard Template Library (STL) and the last in Boost Library published in <http://www.boost.org/>.

2.1.3.1 Comparison results

Each implementation has advantages and disadvantages referred to the runtime and the memory store, generating cases where they are more or less usable, we analyze two commonly operations used to calculate a power graph representation. The first is the

execution of the binary operations between 2 rows of bits, ten thousand times and returns the answer in an index array where each position is equal to 1 in the bit row of the answer, iterations is proportional to the maximum case where the results are significants.

The second is the *And* operation between all pairs of rows (for example if the array length is 250, the number of operations is 62500 or 250^2). The results are showed in table 2.1 and 2.2 respectively, all values are in seconds and the used structure is *Vector*.

Tabela 2.1: Runtime for first operation

Implementation	*Array length*		
	250	*500*	*1000*
Boolean array	159,125	624,525	5035,45
64-Bit intenger array	1,117	5,304	32,361
Bitset	0,017	0,09	0,553
Dynamic bitset	1458	7,586	46,611

Tabela 2.2: Runtime for second operation

Implementation	*time*
Boolean array	17.35
64-Bit intenger array	0.771
Bitset	0.676
Dynamic bitset	1.801

The binary matrix has certain numbers of bytes depending on the implementation, in C++, a vector occupies 16 bytes, but also, we must consider the bytes number for each row. We calculate for rows of 250, 500 and 1000 elements. The results show at Table 2.3 demonstrate that Bitset Is the best option, to stored the binary matrix.

Tabela 2.3: Required bytes for each implemeantion

	Array length		
	250	*500*	*1000*
Boolean array	14016	44016	152016
64-Bit intenger array	13016	42016	144016
Bitset	8016	32016	128016
Dynamic bitset	13016	42016	148016

All evaluations were implemented in Visual Studio 12 C++ and run in Windows 10

PC with a processor Intel(R) Core(TM) i7 of 3.60 GHz and 8,00 GB de physical memory. The results demonstrate that the bitset implementation is the best option for power graphs in runtime and memory.

2.2 Related work

Several works about the search of relevant information into real-world networks have been published in the last years among them, there is a group which investigated networks with clustering algorithms trying to find topological information. Although several clustering algorithms were proposed, others methods for analyzing networks were not deep, especially complete substructure.

In this section, we review the related work divided into three categories: extraction of knowledge from real-world networks (2.2.2), algorithms for power graph analysis (2.2.3), and visualization methods for dense graphs (2.2.4).

2.2.1 Overview

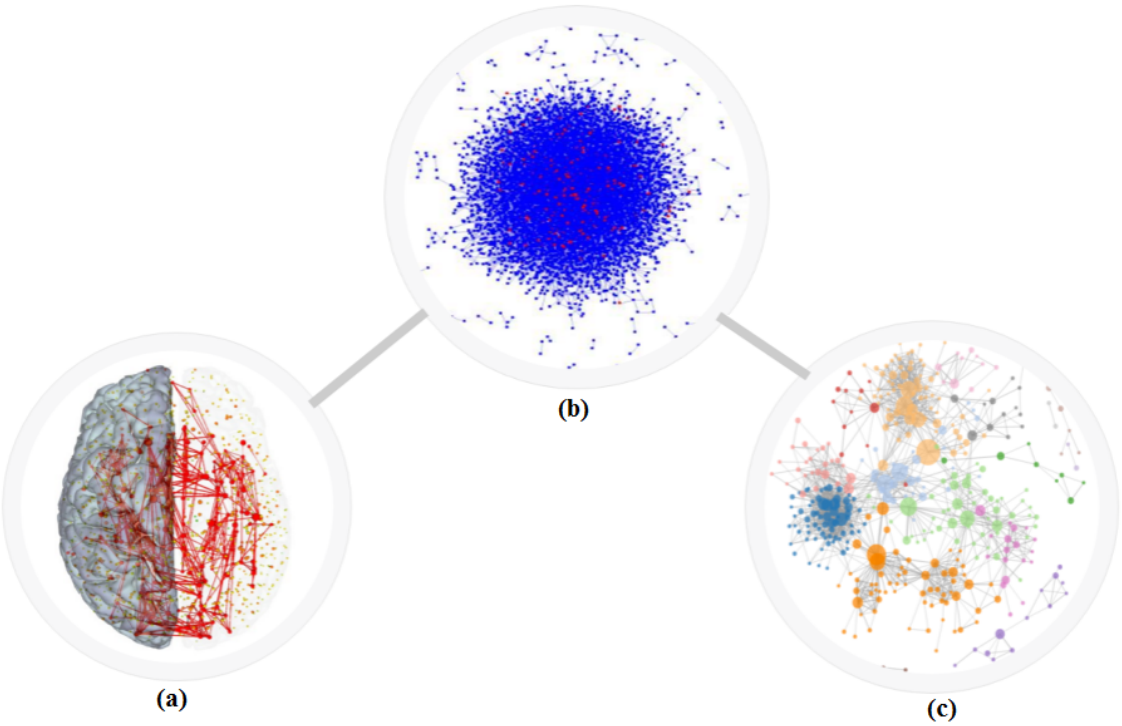
Real-world networks appear from metabolic to social interaction networks. However, their representations have been used in science only in the last century (LEVINSON, 2004), many of them have two main properties: (1) the distance between nodes is short in relation to the number of nodes, and (2) a high clustering coefficient; such networks are also known as small-world networks (WATTS; STROGATZ, 1998). Small-world networks have a hierarchical structure, and for finding them exist algorithms of hierarchical clustering, which require complex diagrams.

2.2.2 Knowledge extraction

In the field of biology and neurology, scientists have studied several real-world networks. In (GAGNEUR et al., 2004) the complex protein purification is described as a graph. After applying a modular decomposition algorithm over this graph, results showed that detected modules corresponded to the reuse in other protein compounds. In (ROYER et al., 2008) a hierarchical graph-based algorithm is used to identify differences in proteins indistinguishable to the naked eye. The relation between drugs and diseases is modeled

as incomplete bi-cliques in (DAMINELLI et al., 2012). The study of brain networks also uses graphs to help understanding different levels of connectivity. Structural connectivity is described as the physical or anatomical connections (SPORNS, 2011). Functional connectivity corresponds to patterns of neural activity usually occurring between distant brain regions (HEUVEL; POL, 2010). Finally, effective connectivity is described as a network of causal effects between neural elements (SPORNS, 2011). The connectome is a map of the connections between neurons and brain regions, and can be defined as a graph, where the brain regions and neurons are nodes and the functional, structural or effective connections are the edges. This kind of graph has several features, such as communities, cores (the group of highly connected nodes, resistant to damage) (HEUVEL; SPORNS, 2013), rich clubs (set of nodes with high centrality) (COLIZZA et al., 2006), and hubs (central nodes of a graph (HEUVEL; SPORNS, 2013)). Figure 2.3 shows networks and their analysis.

Figura 2.3: Examples of real-world networks, (a) connectome and their rich clubs <http://www.newswise.com/articles/highways-of-the-brain-high-cost-and-high-capacity>, (b) Hepatitis C virus infection protein network <http://msb.embopress.org/content/4/1/230>, (c) A sociology citation network and their communities <http://nealcaren.web.unc.edu/a-sociology-citation-network/>.



2.2.3 PGA Algorithms

A power graph algorithm was proposed in (ROYER et al., 2008) for undirected graphs that uses a hierarchical clustering algorithm (EISEN et al., 1998) based on the Jaccard's similarity index (JACCARD, 1901). In (DWYER et al., 2013) it was proposed a heuristic to obtain a decomposition of a power graph with a complexity of $(O(|V|^3 * |E| * log|E|))$. This complexity is high for large and dense graphs such as connectomes. In (DWYER et al., 2014) it is presented another heuristic with constraints on the backtracking algorithm. We propose a bitwise algorithm to find an optimal biclique of the graph. After applying our recursive algorithm, we find a dendrogram of the graph, and eliminate the redundant edges. Our method is different from (ROYER et al., 2008) because we use fast binary operations when finding a biclique.

2.2.4 Network visualization

For large graph visualization, a review of layout algorithms and interactive exploration techniques is given in (HU; SHI, 2015). They recognized challenges of large graph visualization as the problem of dealing with an exponential increase in graph size, the evolution of complex networks, the abstraction and visual representation with novel displays. The orthogonal layout algorithm in (KIEFFER et al., 2016) encodes aesthetic criteria to overcome the deficiency of automatic layout techniques in producing human-like diagrams. The HOLA algorithm achieves layouts of comparable quality to those produced by hand. In (JANKUN-KELLY et al., 2014) it is discussed the state of art scalability, with considerations such as visual limitation, graph design strategies, filtering, aggregation, and interaction. To address the computational consumption of visualizing large networks, (MRžEK; BLAžIč, 2013) developed a GPU-based implementation of a layout algorithm. Using a force-driven method, they generated layouts where communities are less entangled and performed comparisons with the CPU-based implementations. VCD (RUNPENG; JUN; XIAOFAN, 2012) is a network visualization tool used for the analysis of communities and dynamically uncover the hierarchical communitiy structure in real networks. As a use case, they used the *yeast network* and allow the exploration of hierarchical structure by providing a hierarchical evolution function for visual analysis. In (ZHANG; PARTHASARATHY, 2012) they proposed the notion of a Triangle K-core, a simpler topological structure which helps the search for clique-like subgraphs. Addito-

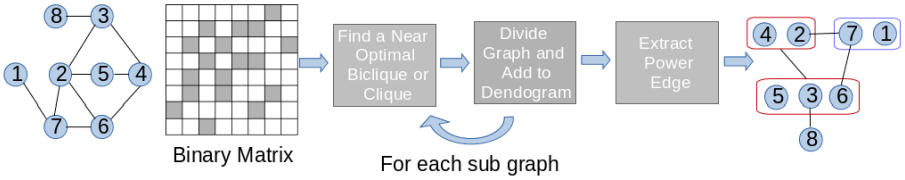
nally, they extend the basic definition of cliques and support user defined clique template patterns with applications to network visualization and correspondence analysis. Dudas et al. (DUDAS; JONGH; BRUSILOVSKY, 2013) presented a visualization which models the overlapping of community membership to solve the problem of members belonging to multiple cliques. They provide an interactive interface to study large network topologies.

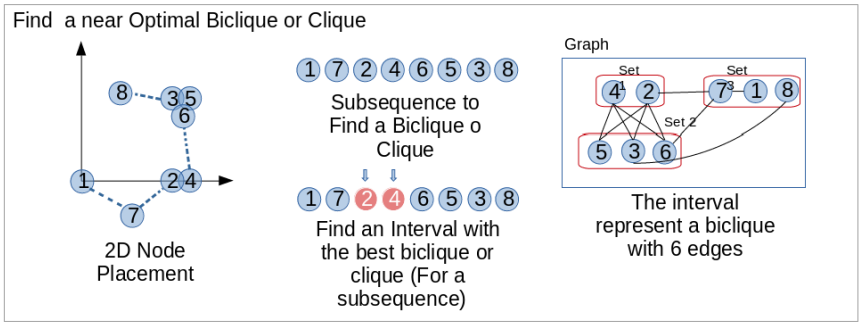
Clustering analysis and matrix visualizations are studied by Wu et. al. (WU; TIEN; CHEN, 2010). HCMapper (MARTI et al., 2015) compare groups extracted from pairs of dendrograms and StructMatrix (GUALDRON ROBSON L. F. CORDEIRO, 2015) a large visualization of dense matrices. In (BECK MICHAEL BURCH; WEISKOPF, 2015) a survey of dynamic graph visualizations is described. They focus on the challenge of representing evolution of relationships between entities. In terms of classifications tasks, (BEAUXIS-AUSSALET; HARDMAN, 2014b) and (BEAUXIS-AUSSALET; HARDMAN, 2014a) describe confusion matrices visualzations to minimize user cognitive effort. (WONG et al., 2013) proposed a matrix visualization for social networks to understand the distances between individuals, groups within the populations and disconnections. In this paper, we use two ways to display clusterings over a power graph. The first one uses a matrix layout that allows to observe how bicliques and cliques groups create clusters. The second one is the orthogonal layout that improves the visualization of individual links and groups of vertices (EIGLSPERGER; FÖSSMEIER; KAUFMANN, 2000).

3 ALGORITHM

In this chapter, we explain the algorithms used to get, display and analyse power graphs. Section 3.1 describes algorithms to find near optimal sub-graphs (biclique, clique or star). Note that the star is a particular case of a biclique and we use the biclique term to refer to both cases. The algorithm to find the dendrogram and redundant edges is explained in section 3.3 and 3.2, respectively. The figure 3.1 has an overview of the first three sections. We show in section 3.5 the process to display power graphs in a quasi-orthogonal layout. In section 3.6 is showed the method to analyse clusters of the network. Finally, the image compression is explained in the section 3.7.

Figura 3.1: Power Graph construction algorithm overview. Input: adjacency matrix representation of the graph. We recursively find optimal maximum edge cliques and bicliques and form groups of nodes. The algorithm builds a dendrogram from previously found cliques and bicliques and finally, abstract reduntant edges in power edges. Output: dendrogram representing the power graph groups.





3.1 Finding Near Optimal Biclique or Clique

Algorithm 1 finds a near optimal solution based on a clustering idea of joining vertices. Vertices that share the highest number of common adjacent vertices are grouped using the Jaccard index as a measure of similarity and projected on a plane in two dimensions. Nodes are sorted according to the Jaccard distance. This heuristic finds an ordering of nodes which might not be optimal; however, we reduce operations as we avoid the search of nodes without common neighbours.

Algorithm 1 Algorithm for finding a near optimal subgraph (clique or biclique)

```
procedure FINDSUBGRAPH(BitMat)
    order ← GETORDERNODES(binMat)
    nearOptimalSubG ← {}
    for i = 0; i < order.length do
        for j = i; j < order.length do
            biclique ← FINDBICLIQUE(i,j)
            clique ← FINDCLIQUE(i,j)
            if Biclique.edges>Clique.edges then
                nearOptimalSubG ← biclique
            else
                nearOptimalSubG ← clique
    return nearOptimalSubG
```

3.1.1 2D Node Placement

The main idea of the power graph algorithm is to find groups of strongly connected nodes and represent them as power nodes. To obtain nodes with the most adjacent nodes in common, we order the nodes based on the number of common adjacent connections. To do so, we characterize each node with the number of adjacent edges and define a distance in 2D space based on the Jaccard distance, calculated as $D_j(A, B) = 1 - |A \cap B|/|A \cup B|$. We multiply by $|A \cup B|$ to limit the result to only integers, then the Jaccard distance equals to $d(A, B) = |A \cup B| - |A \cap B|$.

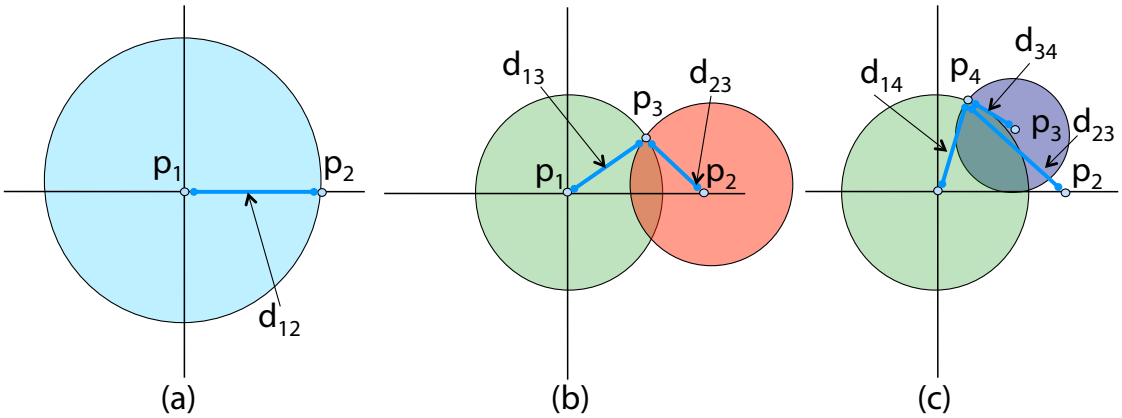
Our iterative approach takes the first node and assigns it to position $(0, 0)$. The second node is placed at a Jaccard distance k of the first node (at $(k, 0)$). Any subsequent node is positioned at the intersection of the circumferences with a radius equal to the two shortest Jaccard distance $d(n_i, n_j)$. Figure 3.2 details the node placement process.

Algorithm 2 collects ideas referred to the above and shows the use of the Jaccard function. The function named *makeCircle* uses the node information and returns a circle where the radius is the Jaccard distance and the center is the node position. The circle intersection has four possible values: zero, one, two and undefined. When the case is 1, 3 or 4 the answer is the value that generates less conflict. The conflict is defined as the sum of Euclidean distances from one node towards all others. Case 2 has only one option.

Algorithm 2 Algorithm for finding the 2D Node Placement

```
procedure POSITIONNODES2DBYJACCARD(bitMat)
    pointList=[]
    firstPoint ← (0, 0)
    pointList.ADD(firstPoint)
    SecondPoint ← (0, JACCARDDISTANCE(1,2) )
    pointList.ADD(SecondPoint)
    for i = 3; i < visit.length do
        list ← []
        for j = 1; j < i do
            list.ADD(JACCARDDISTANCE(i,j))
        frsInd, sndInd ← TWOMIN(list)
        firstCircle ← MakeCircle(frsInd)
        secondCircle ← MakeCircle(sndInd)
        Intersection ← CIRCLEINTERSECTION(firstCircle,secondCircle)
        if Intersection.length = 0 then
            a ← (x_first+JACCARDDISTANCE(i,frsInd),y_first)
            b ← (x_second+JACCARDDISTANCE(i,sndInd),y_second)
            nextPoint ← a ∨ b
        else if Intersection.length = 1 then
            nextpoint ← Intersection[1]
        else
            a ← Intersection[1]
            b ← Intersection[2]
            nextPoint ← a ∨ b
        pointList.ADD(nextPoint)
    return pointList
```

Figura 3.2: 2D Node placement: (a) the first point p_1 is placed at the origin, while the second point p_2 is placed in the x-coordinate axis at the Jaccard distance d_{12} of p_1. (b) the third point p_3 is placed at the intersection of the circles with center at p_1 and p_2 and radius corresponding to their Jaccard distances to p_3 (d_{13} and d_{23} respectively). (c) similarly, the fourth point p_4 is placed at the intersection of the two circles associated to the shortest Jaccard distances to p_4 (in this example the circles defined at p_1 and p_3 with radius d_{14} and d_{34} respectively). The same criterion is applied to subsequent nodes.



3.1.2 Finding the Sequence of Nodes

To compute the ordering of the nodes in 2D, we need to select the nodes with the shortest distance, which represent the nodes with maximal adjacent nodes in common. We formulate the problem of obtaining the sequence of vertices as finding the shortest Hamiltonian path for a complete graph. We use a greedy algorithm with the following steps. First, we take the leftmost node and insert it into the result list, look for the closest node in the Euclidean distance, and add it to the list. The process is repeated until nodes are processed (Algorithm 3). The final list has the ordered sequence of vertices used to find the optimal sub-sequence.

Algorithm 3 Algorithm for finding the Sequence of Nodes

```
procedure GETORDERNODES(bitMat)
    listPoint ←2DNODEPLACEMENT(bitMat)
    leftMostNode ← FINDLEFTMOSTNODE(listPoint)
    order ← []
    ORDER.ADD(leftMostNode)
    LISTPOINT.DELETEPOINT(leftMostNode)
    for i = 1;i < visit.length do
        nextNode ←LISTPOINT.CLOSESTPOINTTO(previousPoint)
        ORDER.ADD(nextNode)
        LISTPOINT.DELETEPOINT(nextNode)
    return order
```

3.1.3 Finding the Optimal Subsequence

In this step, we find the maximum edge biclique or clique computing the number of adjacent nodes in common for each node subsequence. We iterate the ordered sequence with two indices i and j and evaluate for each subsequence the number of common adjacent nodes. For each evaluated subsequence, i represents its initial position in the ordered sequence while j represents the final position. We choose the subsequence which maximizes $(j-i)*(|S|)$, where S is the intersection set of adjacent nodes and $j-i$ represent the number of nodes in the subsequence. Moreover, the pair i, j produces a clique or biclique with the highest number of elements. We choose the pair according to its number of edges. The total complexity to find the approximation of the maximum edge biclique is $O(n^2)$.

3.2 Finding the Dendrogram

A dendrogram has three types of vertices: objects, nodes, and the root node(PHIPPS, 1971). Objects are original graph nodes, the root node and nodes are power nodes but in our case, we also need to distinguish between cliques and bicliques. To solve this, we use different colors for each power nodes in the visual representation: red for bicliques and blue for cliques. The implementation and representation in C++ is explained in the Section 3.2.1.

To represent a hierarchy of node groups, we build a tree-like data structure with the previously found cliques and bicliques. Each of the dendrogram nodes stores a graph node partition with three children, which is analogous to a graph partition in three groups. The first and second child store the first and second sets of the proximal optimal edge biclique respectively. If the second set is empty in case of cliques, this is discarded in the following operations. The third partition represents the remaining nodes of the partition, in others words, it is the difference between the parent and the union between the other partitions.

Algorithm 4 stops when the parent stores two or fewer nodes. The output is a dendrogram structure (see Figure 3.3) with near optimal maximum edge bicliques or cliques considered as power nodes and stored in the variable called *Powergraph*. We also calculate the first power edges of cliques or bicliques taking advantage of the recursion. It can have bad cases, for example, when the subgraph has zero or one nodes and the parent

has a considerable number of nodes. We solved it using a similar method explained in (ROYER et al., 2008). We group nodes and assign power nodes until the smaller partitions have two nodes. It is easy to note that some power nodes are not linked. It is not an important issue and is solved in section 3.3.

Algorithm 4 Algorithm for finding the Dendrogram

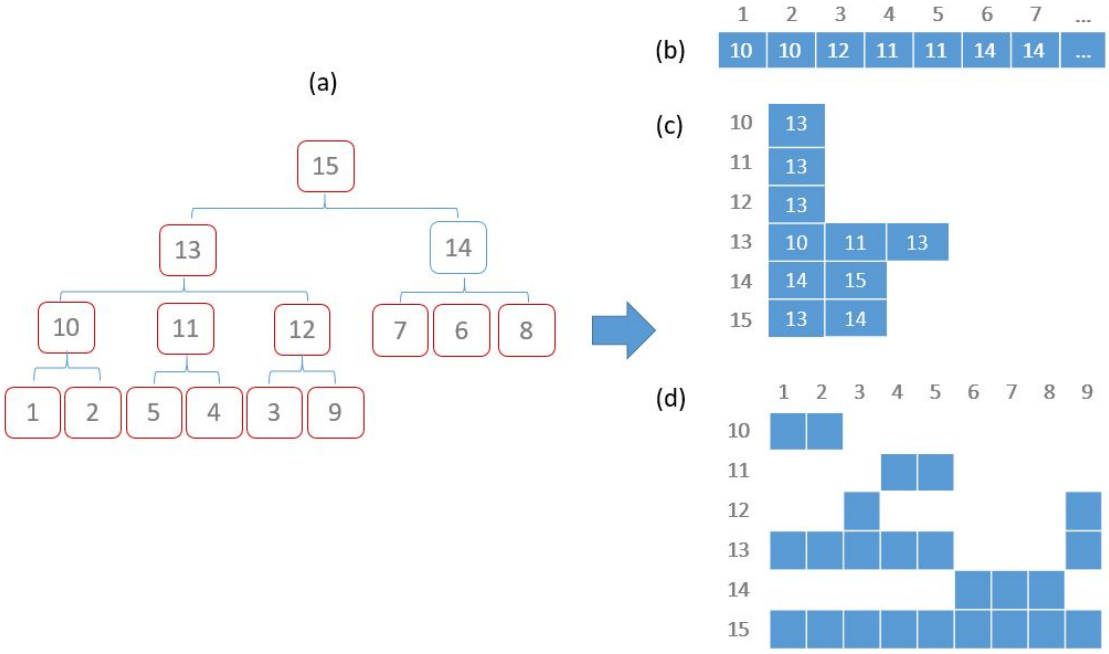
```
procedure POWERNODEASSIGN(bitMat)
    powerGraph ← { }
    if bitMat.nodes ≥ 2 then
        subGraph ← FINDSUBGRAPH(bitMat)
        if subGraph is Clique then
            clique ← subGraph
            if clique.length ≥ 2 then
                Remove the clique of the graph
                Add a power node of the clique
                Add a power edge
                Divide the graph in 2 sets
                repeat PowerNodeAssign for each set
            else
                Look for others possible power nodes
        if subGraph is Biclique then
            biclique ← subGraph
            if biclique.Set1.length ≥ 2 then
                Remove the biclique of the graph
                Add 2 power nodes
                Add a power edge
                Divide the graph in 3 sets
                repeat PowerNodeAssign for each set
            else
                Look for others possible power nodes
    return powerGraph
```

3.2.1 Implementation

A hash array can store trees as dendrograms, and its implementation is easier than traditional trees with pointers, but against the memory usage depends on the maximum index. We adapt the power node indices to have a limit of the maximum index($max_{index} < 4 * n$), starting from $n + 1$ where n is the number of vertices of the original graph. This type of representation is common to see in algorithms bases on Union-find.

Adjacent lists are used to store graphs and trees, thus, we have a direct link to children of a node and the store an edge of one node to itself (as cliques). This representation

Figura 3.3: Implementation of a dendrogram: (a) the dendrogram for a 9-nodes graph, (b) the hash array, (c) the adjacent, (d) Binary rows for each power nodes index.



is extensively used by BFS and DFS algorithms.

A Binary row can store the final children of each power node (see Figure 3.3). In this way, the comparison between two power nodes is direct. We use these ways simultaneously for obtaining a short time in data access and operation that are required in the section 3.3.

3.3 Finding Redundant Edges

The dendrogram structure holds the near-optimal power nodes configuration computed by our method; however, many edges from the original graph need additional processing and to be considered as power edges. We define redundant edges as the individual connections between sets of nodes that belong to power nodes. During this phase, we remove redundant edges having multiple connections from one node to all nodes or connected a power node. All these edges could be represented by a power edge, connecting the node with a power node. We replace them with power edges. Edge reduction could occur with edges going out of from a node or a power node.

Algorithm 5 illustrates these ideas and works in the following way: it requires a specific power node configuration and a graph stored in the variable named *PowerGraph* and *BitMat* respectively. The next step is to visit each node of the original graph and

access its adjacent list of power nodes. The power edge is represented by the pair node and power, as a star, in the last step, it adds the star to power graph and removes from the binary matrix using binary operations.

Algorithm 5 Finding the Redundant Edges

```
procedure POWEREDGEREPLACE(powerGraph,BinMat)
    for each node in BinMat do
        Find Adjacent Power Node
        for each powerNode in adjacentPowerNodes do
            powerGraph.ADDPOWEREDGE({node, powerNode})
            Delete edges from bitMat
```

3.4 Finding adjacent list of power nodes

The idea of Algorithm 6 is to compare two Binary rows with *And* operations. The first are the adjacent nodes corresponding to a node and the second is the final children of a power node, both are parameters stored in *Node* and *PowerNode*, respectively. If the comparison is equal to the power node then it is added to the final solution as a power edge, otherwise, the algorithm is repeated for each child of the power node in a depth-first search.

Algorithm 6 Finding adjacent list of power nodes

```
procedure FINDPOWERNODEADJACENT(bitMat, powerGraph, node, powerNode)
    intersection ← bitMat[node] ∩ powerNode
    if intersection.COUNT() < 2 then
        return {}
    if bitMat[node] = powerNode then
        result.ADD(powerNode)
        return
    get children of the power node
    for each powerNode in children do
        FINDPOWERNODEADJACENT(bitMat, powerGraph, node, powerNode)
    return result
```

3.5 Orthogonal Layout for Power Graphs

Orthogonal layouts are widely used for clear representations of simple graphs. We extend the orthogonal layout idea to visualize power graph diagrams. We propose

a greedy method to compute the position of nodes and power nodes in a 2D plane and compute edge trajectories that avoid overlapping other elements in the layout(Figure 3.4).

3.5.1 Drawing Vertices

The orthogonal layout for power graphs uses nodes as boxes and power nodes as rectangles around the previously positioned boxes. Based on the hierarchy found, we assign a minimal box to each node in the original graph. For each leaf node in our dendrogram, we place the nodes in a bounding rectangle using a clockwise orientation and assign to the position that minimizes the area of this rectangle (Figure 3.4).

We calculate the bounding rectangle layout by ordering the boxes from highest to lowest area. The first box is assigned to position $(0, 0)$ in the plane. Subsequently, the placement of the following nodes is chosen from four possible positions around the bounding box of previously located boxes (up, down, left, right). We choose the position where the value of F is minimized:

$$F(h, w) = h * w * |h - w|$$

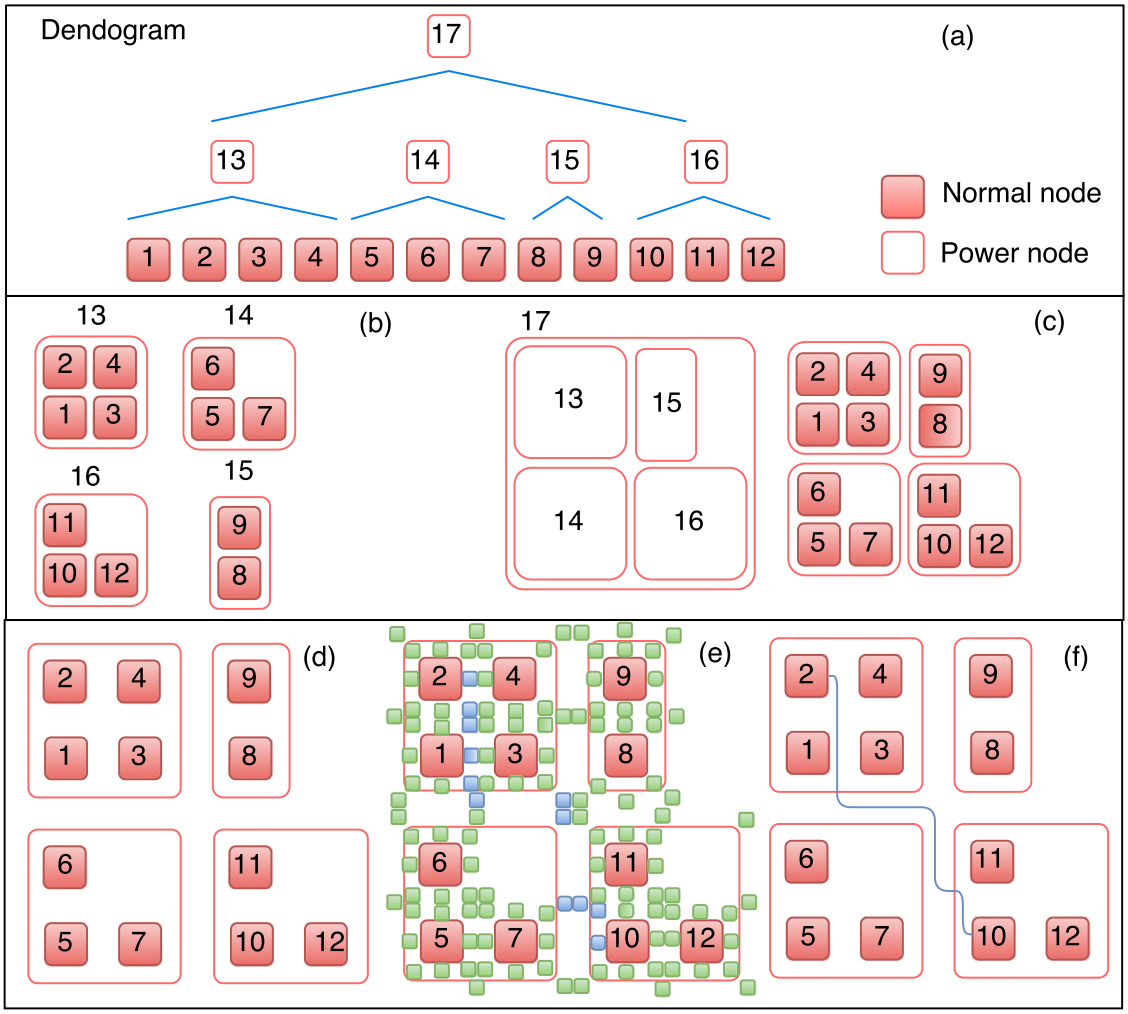
where h and w are the height and width of the minimum bounding box that contains the previously placed boxes.

3.5.2 Drawing Edges

We use a roadmap approach to place edges. The graph can be seen as a city, where nodes and power nodes are buildings, and blank spaces in the layout represent two-way roads where we place graph edges. To draw an edge, we need to select a route between two nodes that does not cross other edges. In addition, the direction of the edge must be parallel to the coordinate axes. The path of an edge must be minimized to produce a less cluttered visualization.

To calculate the shortest path distance between nodes, we use articulation points (Figure 3.4) which represent possible ways for edge drawings. Each articulation point is represented as a node in a helper graph and edges describe the adjacency between articulation points. We use a helper graph to find the shortest distance between nodes in the original graph. Edges are colored in blue and the opacity of the edges is mapped to

Figura 3.4: Power graph orthogonal layout algorithm. (a) dendrogram representation of power graph. (b) We begin node positioning from the last level of the dendrogram and sort squares by area. (c) Inner nodes are represented as bounding rectangles. (d) We place the power nodes (bounding rectangles) in clockwise fashion after ordering them by area. (d) node layout with additional spacing to allow room for edge connectivity (e) green articulation boxes added, along blue boxes that represent a path an edge should take to connect node 2 to node 10. (f) edge path connecting the selected articulation boxes.

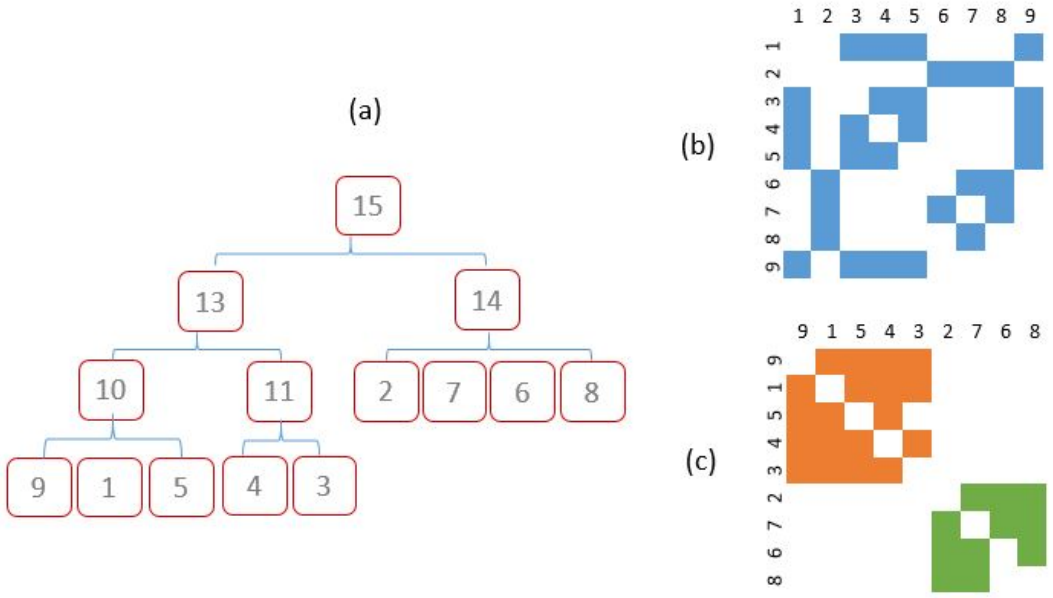


the number of edges passing through the same track.

3.6 Clustering using Power Graphs

Given a large graph, we want to use the properties of power graphs to propose a user-guided clustering method. The BCD algorithm finds groups of nodes based on the number of common adjacent neighbors. Thus, the ordering of nodes is reflected in the dendrogram structure. For the visualization, we choose an adjacency matrix to avoid edge overlapping and demonstrate the change in the order. The rows and columns of the

Figura 3.5: Power graph orthogonal layout algorithm. (a) . (b) squares by area. (c) Inner nodes are represente



adjacency matrix (Figure 3.5) are ordered by the number of common neighbors represented in the dendrogram. We built an interactive user interface that allows the user to visually select the clusters in the matrix. Clusters are represented by squares of the same color located on the diagonal of the matrix and the colors in the visualization matrix are mapped to properties of the graph nodes.

3.7 Image reduction

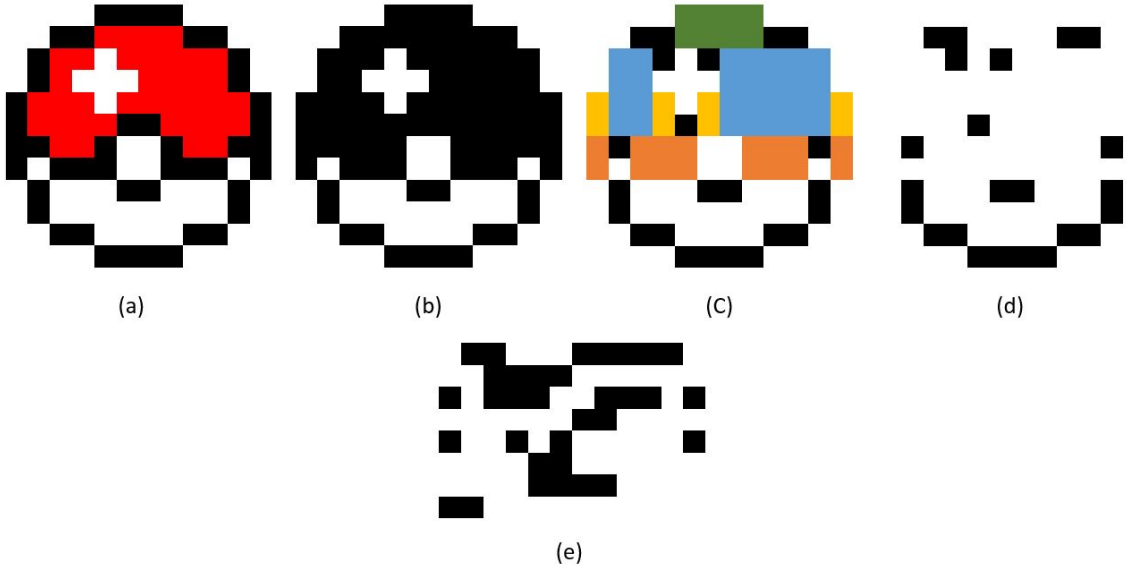
The digital communication is a continuous need with an evolving situation, particularly to images and videos. In this sense, the data compression is involved in several works. Image compression is an extensive field of research, it may be lossy as BMP or PNG or lossless as JPG, this last is the most commonly used for image transmission.

We propose a novel lossless method based on biclique detection. The idea behind Algorithm 7 is to interpret the binary image as a bipartite graph, extract bicliques and replace the edges with a simple coding.

The decoder receives a binary image and a number, where each single pixel of the binary image is one or zero. This number indicates how many times the algorithm for finding bicliques is executed. Bicliques are removed from the binary matrix and eventually are saved as binary rows, exactly two rows for each biclique (see Figure 3.6).

A biclique represents $|S_1| * |S_2|$ pixels graphically, but it only needs $|S_1| + |S_2|$.

Figura 3.6: Image compression based on biclique detection: (a) full-color image, (b) binary image, (c) detected bicliques in the binary Image, (d) binary image without bicliques, (e) biclique binary image.



Depending on the length of S_1 and S_2 the reduction is considerable, for example, if $|S_1| = 15$ and $S_2 = 20$ graphically they represent 300 pixels but they only require 35 pixels, in this case, the reduction is more than 80%. To decode the biclique and reconstruct the image, it requires a simple iteration for each biclique.

Algorithm 7 Binary image compression

```
procedure IMAGECOMPRESSIONENCONDER(BinaryImage,Iterations)
    bitMat ← BinaryImage as binary matrix
    bitMatBiclique ←
    for i = 0; i <Iterations do
        find biclique
        delelte biclique of bitMat
        save biclique in bitMatBiclique
    return bitMat, bitMatBiclique
procedure IMAGECOMPRESSIONDECODER(bitMat,bitMatBiclique)
    for each biclique in bitMatBiclique do
        S_1, S_2 ← biclique
        for each node_i in S_1 do
            for each node_j in S_2 do
                add edge (node_i, node_j) in bitMat
    image ← bitmat as binary image
    return image
```

4 RESULTS

In this chapter, we report our results with respect to the performance of the BCD algorithm, as well as the orthogonal layout and clustering in three different datasets and image compression results.

4.1 Synthetic and Real Datasets

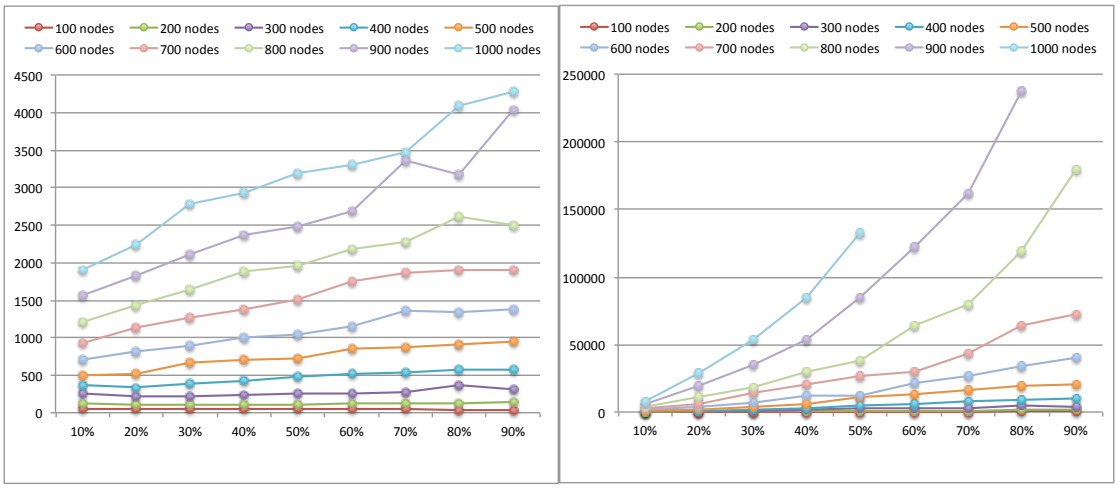
A synthetic dataset of 90 random graphs was created as follows. The density was defined to be between 10% and 90%, with an increment of 10%. The number of vertices in the graph started at 100 nodes, with increment of 100 nodes until the last graph with 1000 nodes. All edges have an equal probability of appearing in the random graph. These features provide an equitable scenario for PGA algorithms.

We used three real datasets: the cat cortex connectome, the zoo dataset and a dermatology dataset. The cat cortex dataset contains information about brain regions of the cat cortex. We build a graph from the dataset where each graph node represents a brain cortex region and edges represent the existence of functional connectivity between cortex regions. The zoo dataset contains animals with 17 boolean valued attributes. The constructed graph uses animals as nodes and represents the similarity of animals with edges. Finally, the dermatology data corresponds to 365 patients with clinical attributes. Each patient has one of six dermatology diseases.

4.2 BCD Performance Results using Synthetic Datasets

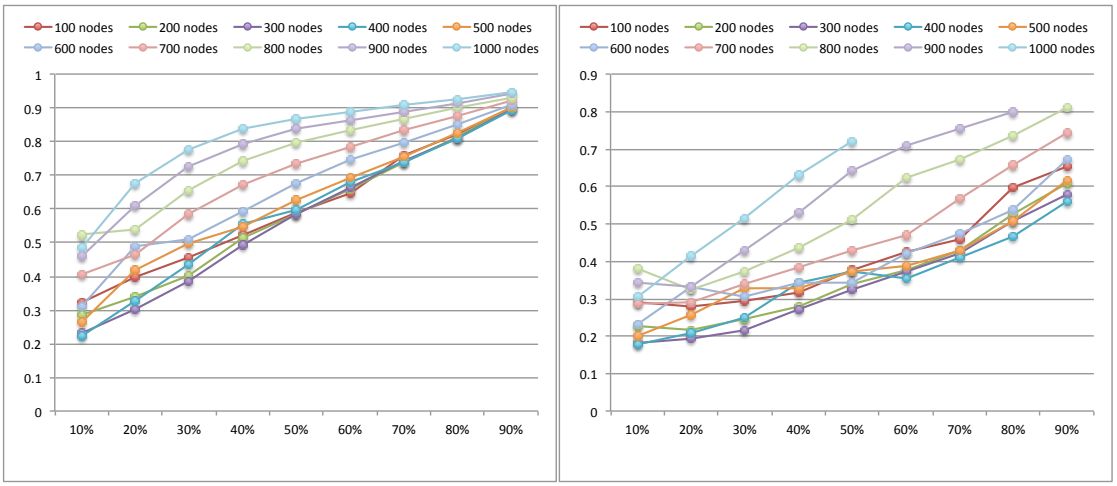
We compared our algorithm against the work in (ROYER et al., 2008) using the synthetic dataset. BCD shows good performance with different configurations, which is expected in real word applications. The runtime of our algorithm shows a clear advantage over previous related work (Figure 4.1). The previous algorithm has problems with graphs of 900 nodes with 90% of density and with graphs of 1000 nodes with 60%,70%, 80%, and 90%. For those cases, the java implementation by (ROYER et al., 2008) has runtime errors of memory limits. the edge reduction ratio, our algorithm also has an advantage over previous work. We achieved an edge reduction ratio of 90% for very dense graphs.

Figura 4.1: Runtime performance and edge reduction comparison. Colors represent different graph configurations with certain number of nodes. Runtime in miliseconds and edge d and density obtained by the greedy method (ROYER et al., 2008) (a) and (c) and BCD (b) and (d).



(a) BCD performance

(b) Greedy performance(ROYER et al., 2008)



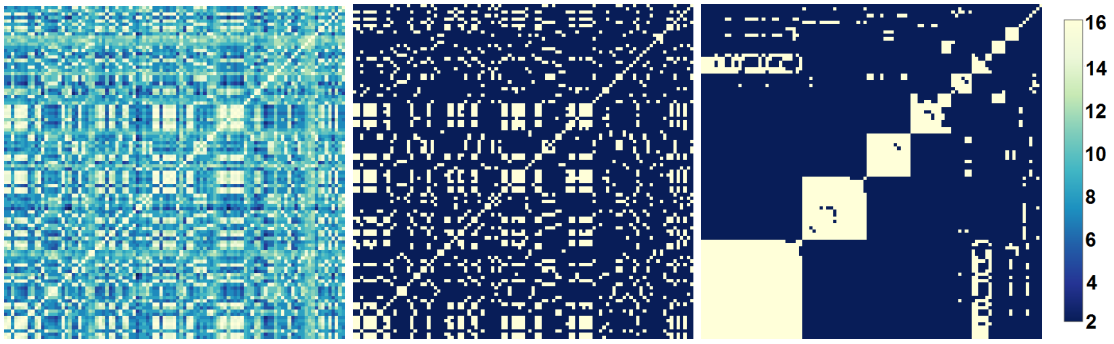
(c) BCD edge reduction

(d) Greedy edge reduction(ROYER et al., 2008)

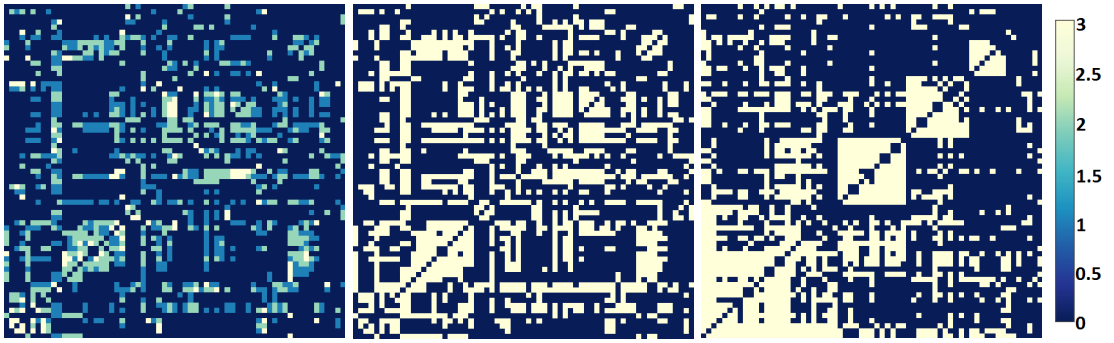
4.2.1 Clustering Evaluation using Real Datasets

We selected real datasets to evaluate the clustering approach using power graphs, Additionally, we compare them with the ground truth datasets and evaluate the accuracy of the technique using confusion matrices (see Figure 4.3). A requirement of our approach is that the input graph must be undirected. We transform the input graph as follows. First, we generate a weighted adjacency matrix – the cortex cat dataset is provided in this format. Then, we compare each pair of elements and assign a weight according to their similarity. We define a similarity distance to compare attributes, normalized between 0 and 1. As an example, consider the animal dataset: the similarity value between breast

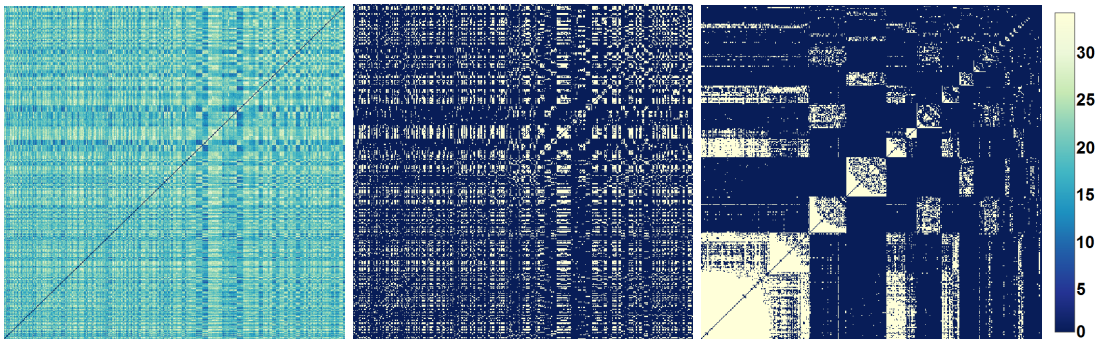
Figura 4.2: Adjacency matrix visualization with columns and rows ordering for three datasets. From left to right. Weighted adjacency matrix. Binary adjacent matrix. Binary Adjacent matrix with rows and columns ordered by maximum edge bicliques.



(a) Animal dataset



(b) Cat connectome graph



(c) Dermatology dataset

features is 1 if the two animals have breasts, otherwise it is 0. We consider that each attribute can be quantitative or qualitative; in case of quantitative attributes we consider the following formula:

$$C_{i_{(A,B)}} = 1 - \frac{|C_{i_A} - C_{i_B}|}{C_{i_{max}} - C_{i_{min}}}$$

Where C_{i_A}, C_{i_B} are values of the attribute C_i of elements A and B respectively and $C_{i_{max}}, C_{i_{min}}$ are the maximum and minimum values respectively for that attribute. For a qualitative attribute it is considered 1 if they are equal and 0 if they are different. Finally, the similarity between two elements is given by:

$$W_{(A,B)} = \sum_{i=1}^{|C|} C_{i_{(A,B)}}$$

where $|C|$ is the number of attributes of an element. Once the weighted matrix is computed, we use a threshold to compute the adjacency matrix binarization. We highlight the importance of considering a suitable threshold due to the fact that the density of the resulting matrix impacts on data analysis. For each dataset, we picked the threshold which generates a density of 50%. A lower density could present a loss of information and a higher density would include information not present in the original graph (see Figure 4.2).

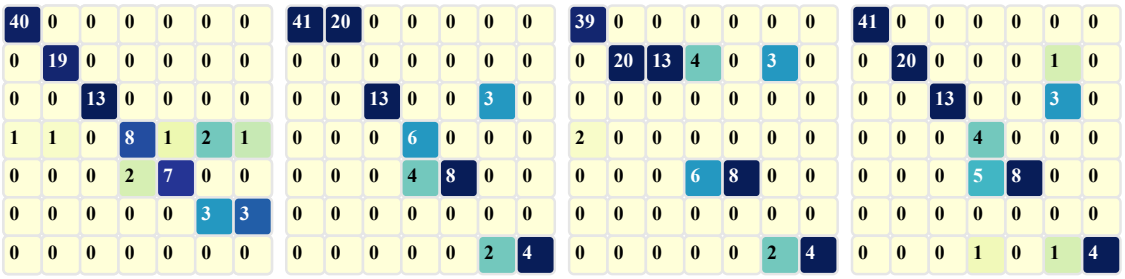
4.2.1.1 Cat Cortex

We show the comparison between the standard adjacency matrix of a cat connectome with the adjacency matrix with ordered nodes in Figure 4.2b. The cat connectome has four regions with high connectivity (communities)(REUS; HEUVEL, 2013). These anatomical regions correspond to the clusters found by our power graph clustering approach. Our method presents a fifth cluster with elements that apparently do not correspond to a defined community. Detected communities have a similarity with a cat brain cortex described in (REUS; HEUVEL, 2013). The first cluster corresponds to the frontolimbic cortex, the second with the auditory cortex,the third with the visual cortex and the fourth with the somatomotor cortex. We should note that the second community which corresponds to the auditory cortex is a quasi-clique. It is well known that cats audition is one of the best in the animal kingdom, thus, we believe that the relationships between animal capacity and density of brain regions are probable.

4.2.1.2 Zoo animals

The classification of elements allows us to understand the behavior of groups of elements.There are different ways of classifying the animal kingdom (taxonomy). We tested the results of the groups obtained with our BCD algorithm against the natural grouping of 101 species of animals. A feature vector of 17 boolean animal characteristics and a similarity matrix of animals features was defined. The matrix consists of similarity values between animals considering the number of characteristics in common. We binarized the matrix considering a similarity of over 13 characteristics to consider, which are repre-

Figura 4.3: Confusion matrices for clustering methods. From left to right. Our approach using power graph, Hierarchical clustering, Kmeans clustering and Partitioning around medoids. The confusion matrix visualization allows an overview of the performance of the algorithms. Each column of the matrix represents the instances in a predicted class while each row represents the observations in the class specified by the classifier. Colors are mapped from 0 to 1 from the normalized confusion matrix.

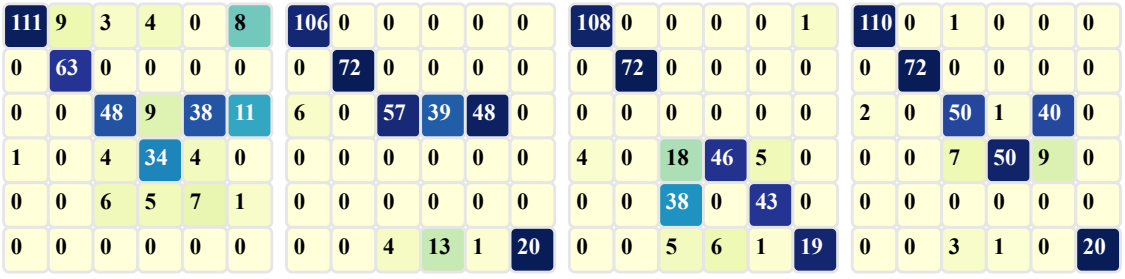


Accuracy: 0.8911 Accuracy: 0.7129 Accuracy: 0.703 Accuracy: 0.8911

(a) Animal dataset

Accuracy: 0.6923 Accuracy: 0.6462 Accuracy: 0.7385 Accuracy: 0.6462

(b) Cat connectome graph



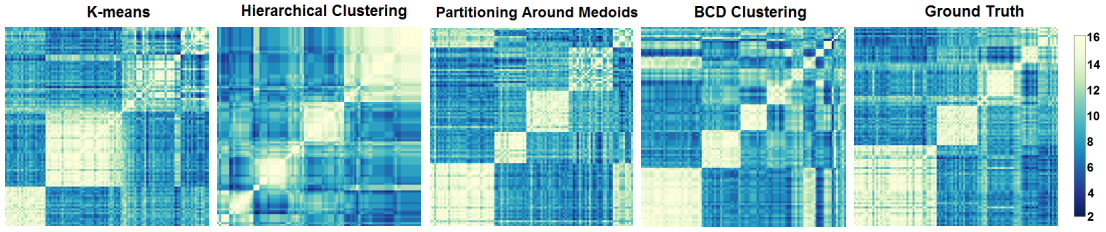
Accuracy: 0.7186 Accuracy: 0.6967 Accuracy: 0.7869 Accuracy: 0.8251

(c) Dermatology dataset

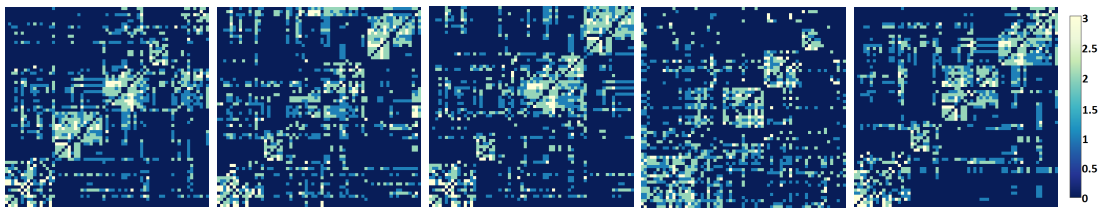
sented as edges (see Figure 4.2a). The threshold was chosen because it generates a graph with high density. The power graphs of dense graphs tend to capture relevant information about connectivity, whereas low-density graphs have many small non-relevant groups.

In Figure 4.2a, the matrix was ordered following our power graph approach. White square regions represent groups and semi-white regions represents links between groups. The adjacency matrix helps us explore the graph and identify the occurrence of groups with high connectivity, thus representing a similarity between their members. Using this concept we can create a simpler graph than the original, using power nodes as simpler nodes. The graph representation was implemented using the D3 library (BOSTOCK;

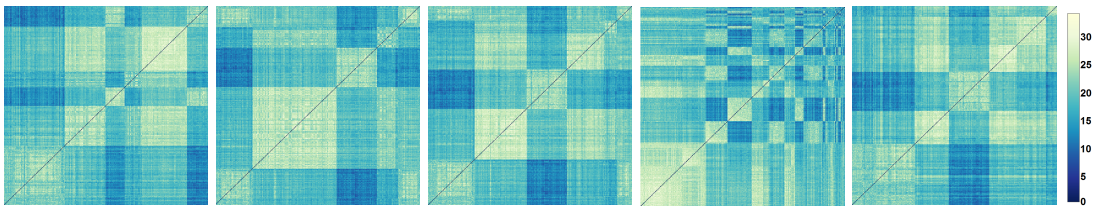
Figura 4.4: Weight adjacency matrix for different clustering algorithms. From left to right, k-means, hierarchical clustering, partitioning around medoids and ground truth.



(a) Matrix layout for Zoo dataset



(b) Matrix layout for Cat connectome dataset



(c) Matrix layout for dermatological dataset

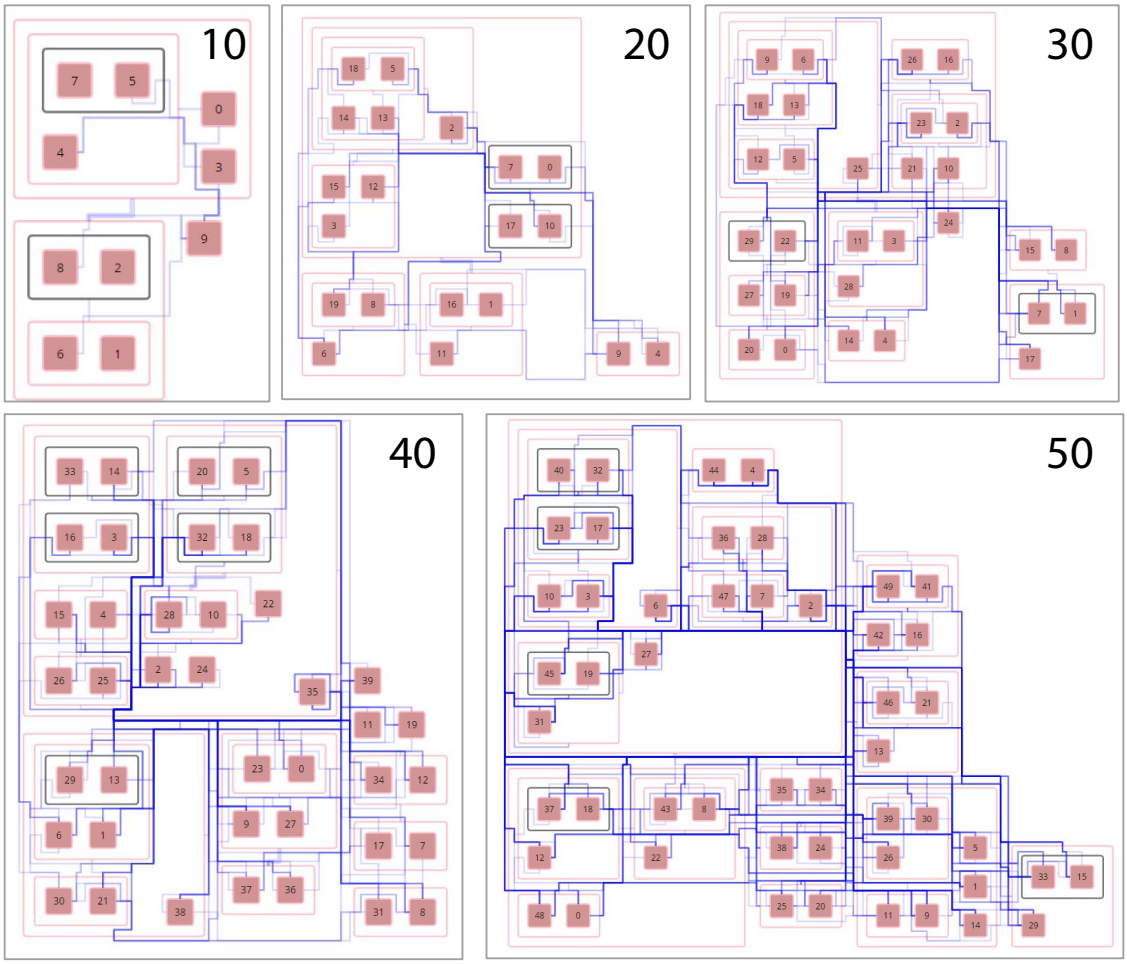
OGIEVETSKY; HEER, 2011). The matrix layout represents a compact way to visualize clusters and the orthogonal layout reduces the edge overlapping in the graph. The proposed layout makes it possible to inspect the difference between cliques and bicliques. Cliques are represented by boxes with black borders and bicliques by pairs of pink boxes connected by a power edge.

4.2.1.3 Dermatological Diseases

The dataset of dermatological diseases corresponds to 365 patients and each patient has 34 attributes divided into two groups, clinical attributes and histopathological attributes. The dataset has 33 integer values and one of them is nominal (LICHMAN, 2013). The numerical attributes were given a degree in the range of 0 to 3. The value 0 indicates that the feature was not present, the values 1,2 for relative intermediate values and 3 for the largest amount possible. The six classes describing dermatological diseases are: psoriasis, seborrheic dermatitis, lichen planus, pityriasis rosea, chronic dermatitis, and pityriasis rubra pilaris 4.2a.

To assess the dermatology data as with the previous datasets, we chose three po-

Figura 4.5: Power Graph orthogonal layout for dense random graphs with 10, 20, 30, 40 and 50 nodes.



pular clustering methods to compare their accuracy and results with our clustering approach. The disadvantage of unsupervised techniques as clustering analysis is difficult to assign labels to discovered clusters. To overcome the problem we used the ground truth and assigned a label based on the more frequently observations inside detected clusters. The precision of our method compared with other clustering techniques are shown in Figure 4.4.

The Circular layout of dendrograms for cat connectome, dermatological diseases and zoo dataset are shown in Figure 4.7. We see the power node indices and the tree structure with branches and colors associated with each cluster.

4.2.2 Orthogonal Layout Evaluation

In addition to the real datasets, we analysed the orthogonal layout in five undirected random graphs with a different number of nodes densities. The main concern of our layout algorithm was the area required for each power graph representation. We can see in Figure 4.5 that an increment of the number of nodes does not generate edge occlusion; however the difficulties to interpret the visualization arise when we want to identify the path of single edges. The proposed orthogonal layout helps us to observe a concentration of edges and their communication with groups of nodes. We considered as a unit of measure the width and length of a single node (represented by a square). In this layout, if the diagram minimises the visualization area, the output layout is more compact and easier to understand and represent more information.

The 10-nodes graph has two small cliques, a large biclique, two dependent groups and ten edges; the original graph has 36 edges which lead to an obvious reduction of more than half edges. The other cases show similar results highlighting the modular structure and power edges that are similar to veins in the circulatory system with orthogonal directions.

In Figure 4.6 we present the orthogonal layout with clustering highlights for the zoo and dermatology datasets, the cat connectome is in Figure 1.1d.

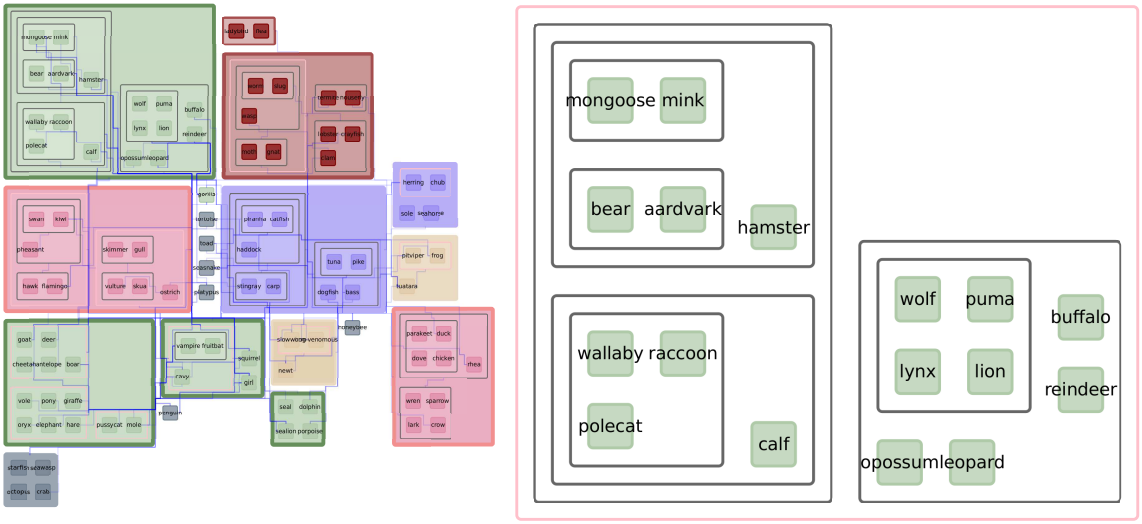
4.3 Image comprenssion performance

We tested the compression method in three images of different sizes. The first is a picture of 256x256 pixels with a density of 51.38 % (see Figure 4.9) where rectangular shapes dominate and large bicliques are easy to find. The second is a widely-known image in the image processing area. Lena is a 512x512 image(see Figure 4.10), with a low pixel density(37.71%), in this case, the first bicliques are not large. Lastly, the third image has a size of 1024x1024 and represents a landscape with a higher density than the others(61.12%) but without rectangular shapes (see Figure 4.11).

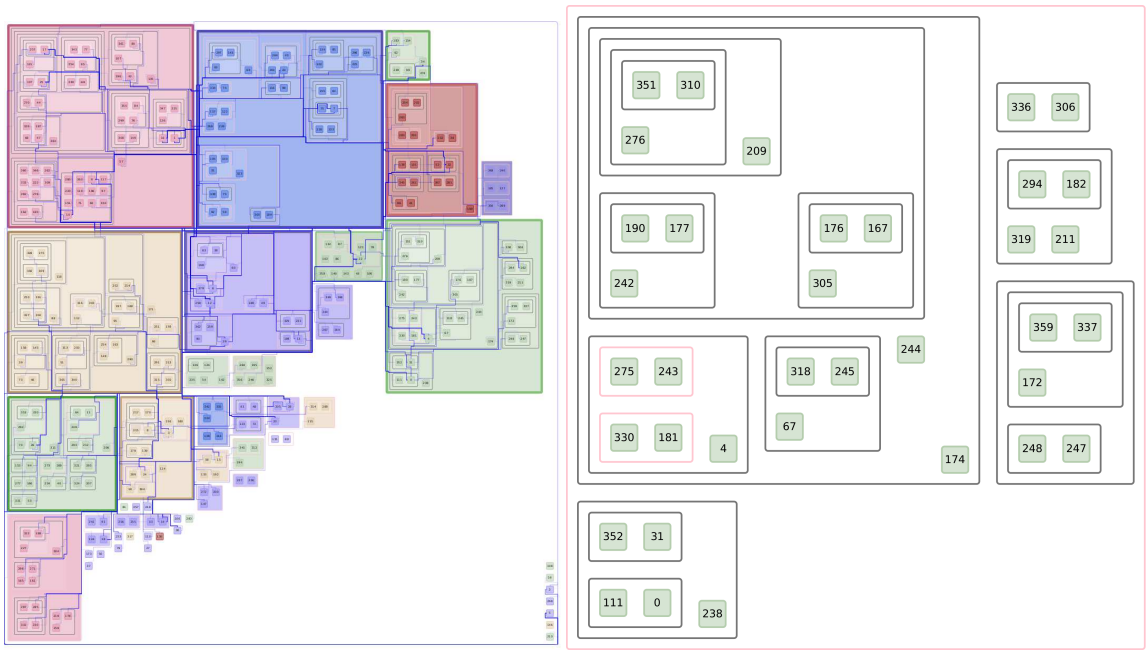
The results show that the pixel reduction ratio depends on three factors: the black pixel density, the rectangular shapes and number of bicliques. In this sense, the images with high density and rectangular shapes have a considerable reduction.

Another method to compress binary images works with 16x16 blocks similar to the JPG strategy and finds rectangular partitions of the same colour (FALKOWSKI, 2004).

Figura 4.6: Power Graph Orthogonal Layout, each box represent a node or power node and their colors correspond to clusters find by BCD method. Generally there are pairs of boxes with the same color (they are bicliques), also there are cliques but they generally are small. in very dense graphs, this is the opposite.



(a) Animal dataset, on the left it shows the complete power graph and on the right a subgraph belonging to the cluster of mammals.

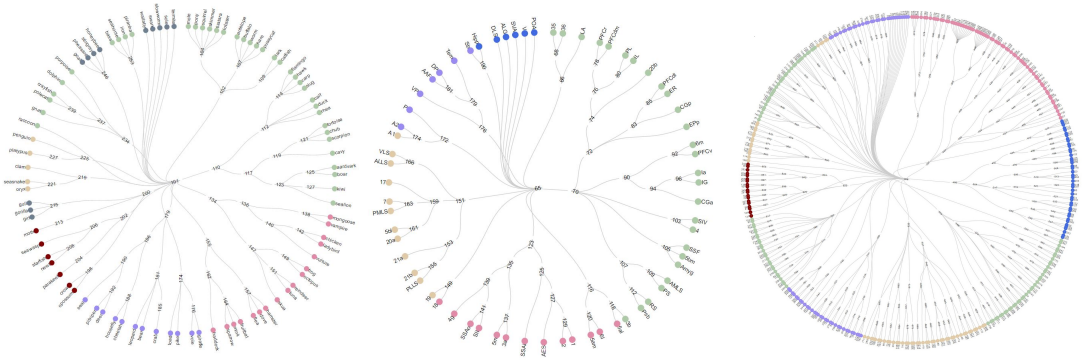


(b) Dermatology dataset, on the left it shows the complete power graph and on the right a subgraph belonging to the cluster of psoriasis patient.

In (MOHAMED; FAHMY, 1995) nonoverlapping rectangular regions are calculated. In conclusion, the plan to compress binary images is to find rectangular regions. The rectangular region is a particular case of a biclique but we can find several rectangular regions in a single biclique (i.e. in Figure 4.11 for $K = 1$ this image have several rectangular regions where K is the number of bicliques).

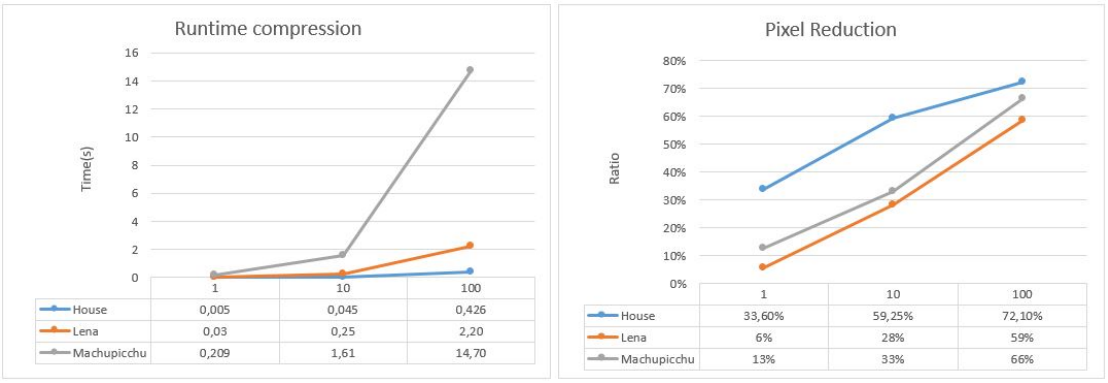
Our method allows applying another method of compression because we only

Figura 4.7: Dendrograms for power graph representations. Leaf nodes represent nodes from the original graph which are colored according to found clusters. Intermediate nodes present tags that correspond to power node identifiers. The arragement of the ramifications from the dendrogram root is the ordering obtained as a result of BCD algorithm.



(a) Animal dataset (b) Cat connectome (c) Dermatology dataset

Figura 4.8: Runtime and black pixel compression for 3 images of different sizes.



reduce the number of black pixels. In others words, it is another way to see the image.

The runtime of our method is particularly fast to work with images from 256x256 to 1024x1024 without using blocks (see Figure 4.8a). The biclique representation also demonstrates that the compression of black pixels is considerable(see Figure 4.8b).

Figura 4.9: 256x256 image with rectangular shapes and 33674 black pixels. For one biclique ($K = 1$) the number of black pixels is 22359 with a compression of $33,60\%$. For $K = 10$ and $K = 100$ the image has 13721 and 9365 black pixels respectively.

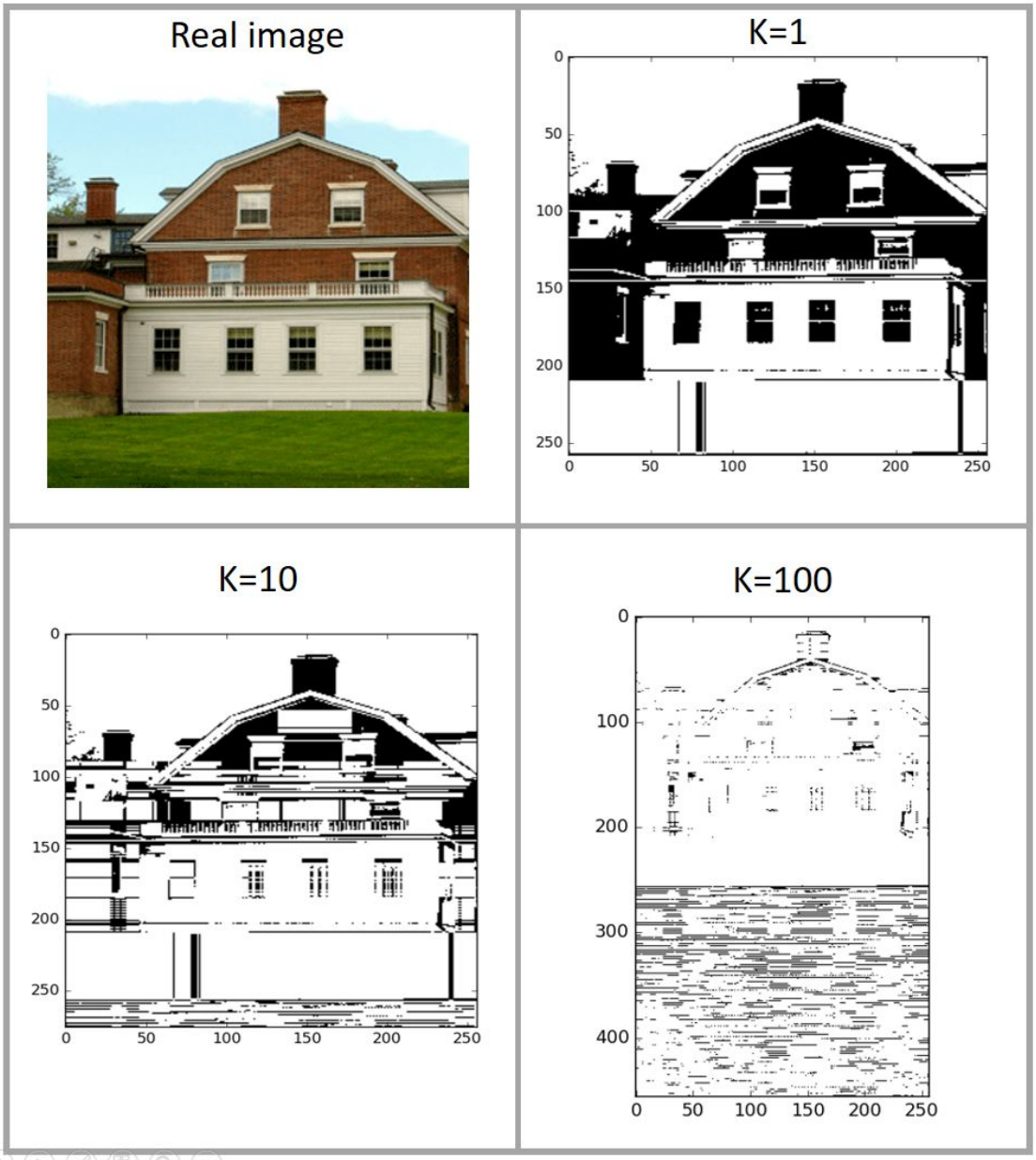


Figura 4.10: 512x512 image without rectangular shapes and no dense, it has 98879 black pixels. For one biclique (K=1) the number of black pixels is 93423 with a compression of 6%. For $K = 10$ and $K = 100$ the image has 70985 and 40802 black pixels respectively.

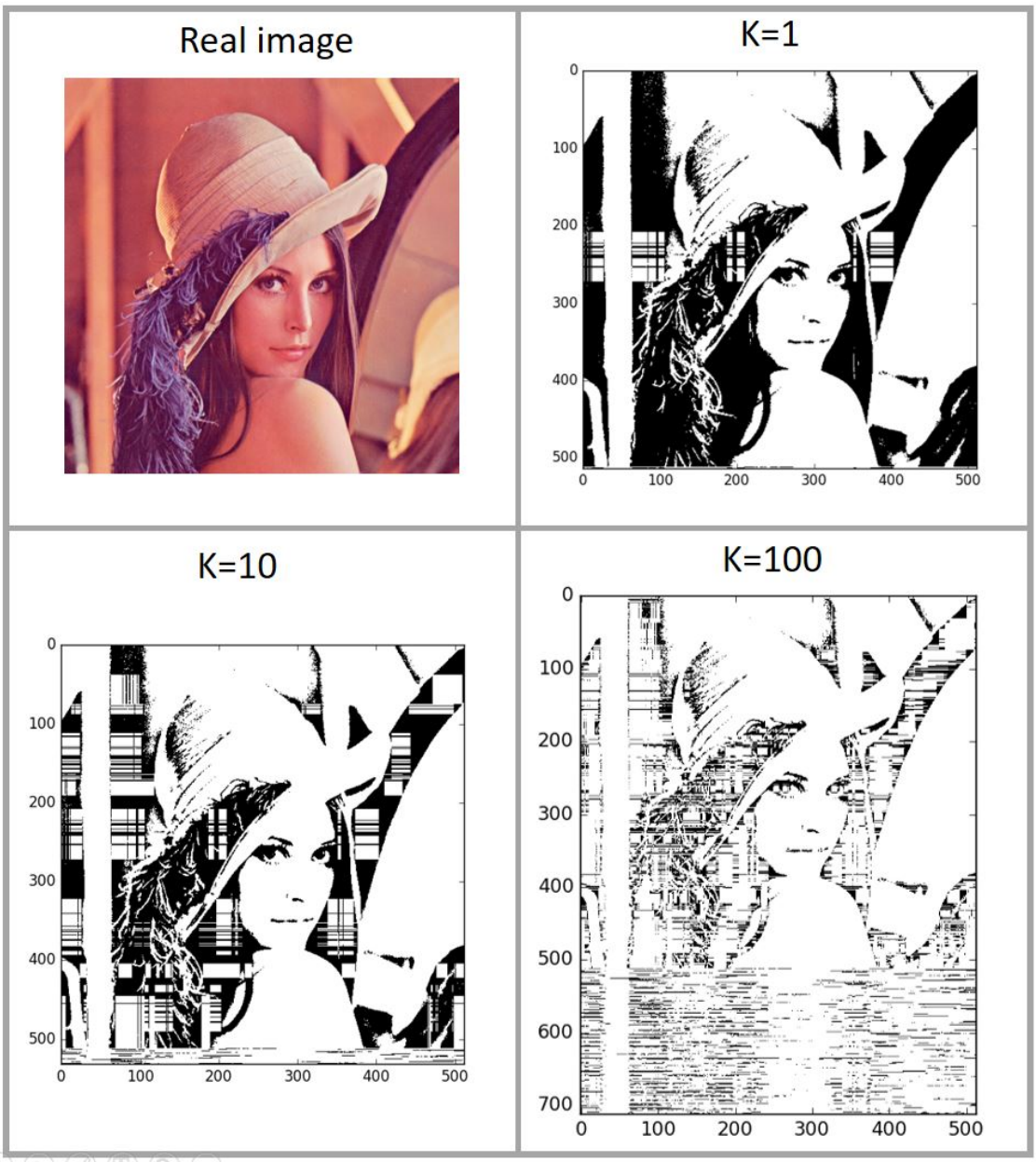
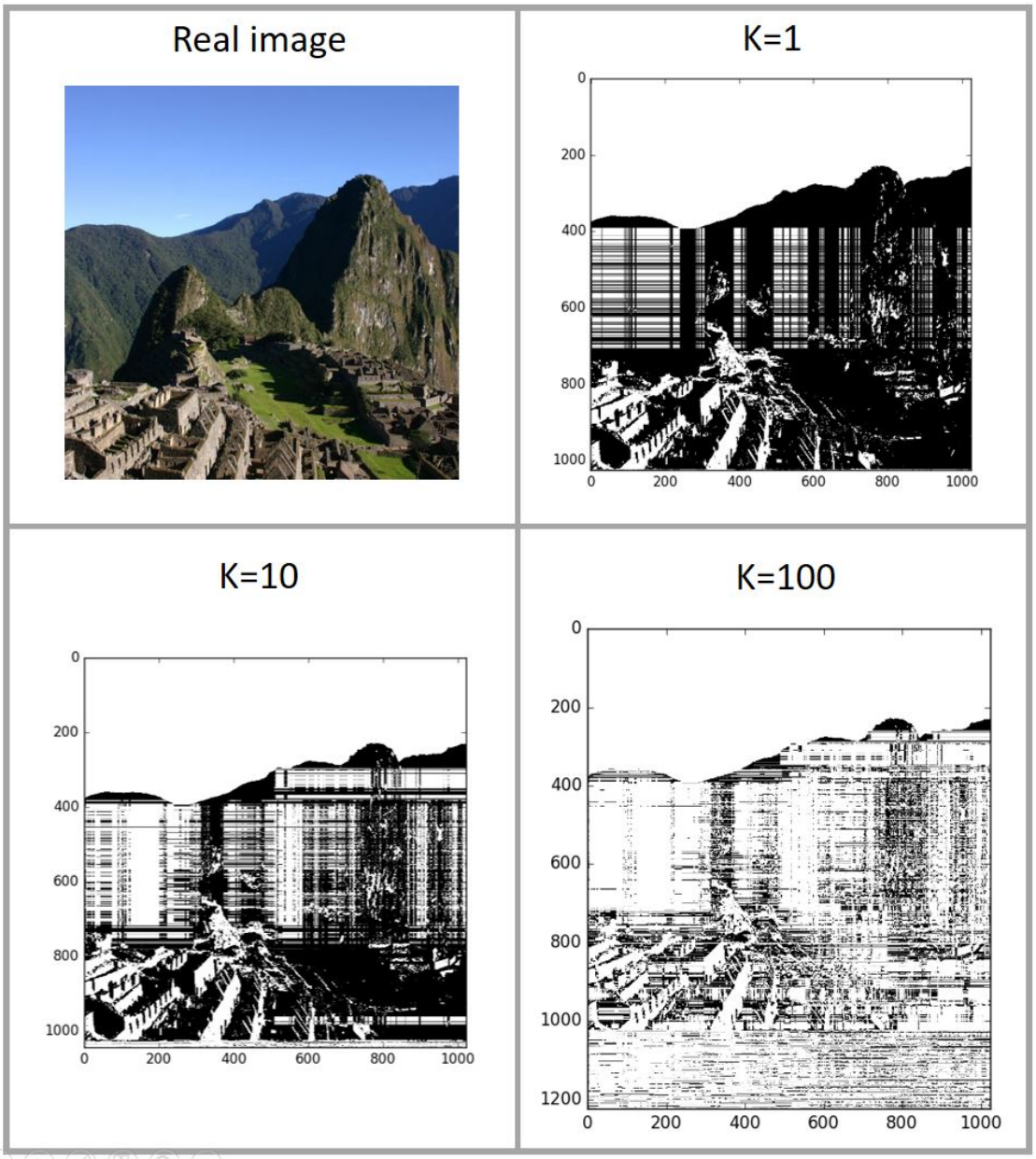


Figura 4.11: 1024x1024 image without rectangular shapes, it has 641900 black pixels. For one biclique (K=1) the number of black pixels is 561303 with a compression of 13%. For $K = 10$ and $K = 100$ the image has 430840 and 215764 black pixels respectively.



5 DISCUSSION

In order to offer more effective methods for data and graph analysis, we designed the BCD algorithm for power graph construction which improves the complexity of related work algorithms and increases the runtime performance over undirected graphs. Additionally, we proposed a clustering technique based on an adjacency matrix visualization with nodes ordering. We evaluated our technique in three aspects: runtime performance, clustering results and visualization presentation. We measured the performance of the proposed algorithm and observed speedups in runtime with respect to the current state of the art. The BCD algorithm possesses an advantage over its rivals during the sets operations while computing common adjacent nodes. Greedy algorithms for power graph construction are required to test each node versus the rest of them resulting in a complexity of $O(n^2)$ where n is the number of nodes of the input graph. Additionally, previous work needs to iterate over each adjacent node to compute the common neighbors. BCD reduces the complexity of finding common neighbors by storing the neighbors as binary sets and computing the intersection with binary operations. For edge reduction, our approach obtains its advantage by maximizing the number of nodes that are abstracted in a power node. We should highlight the limitation of BCD while dealing with larger graphs with more than 1000 nodes; however, the scalability improvement achieved with our method opens the opportunity for real-time graphs applications.

Case studies demonstrate that BCD clustering can be used for clustering tasks supported by the user. The accuracies obtained with our power graph clustering method are similar to results obtained with common clustering techniques for the animal (89%) and cat connectome (69%) dataset. We attribute the difference of accuracies observed in the dermatology dataset to the distance used to form the groups. We used a generic distance for each dataset which measures the number of similar features between the observations. In the case of a health dataset, the distance used does not capture the relevant features to diagnose a disease. However, using the generic distance after the binarization of the adjacency matrix, the generated graph was dense enough and our power graph algorithm obtained interesting groupings that could be used for medical decisions when evaluating a disease.

The orthogonal layout visualization used for power graph representation, solved the problem of edge occlusion with traditional graph visualizations. Additionally, the chosen representation improves the visualization of nodes due to the introduced order.

Displaying the nodes with a natural or custom order help experts in finding patters through data analysis. We tested the orthogonal layout using our three datasets and colored the power nodes with clusters found by our approach. We showed that the proposed layout highlights relationships between subgraphs and subcomponents. In the other hand, it presents the disadvantage of recognizing individual connections, thus allowing a general analysis which could be improved by additional interaction.

The binary image compression is a premature technic, but its results are promising. we demonstrate that our method for finding bicliques is good,but its implementation can be improved with different methods. The black pixels dense determinate the percentage of compression, but we obtained results over 50% of compression for no dense images.

6 CONCLUSION AND FUTURE WORK

In this work, we proposed a novel algorithm for power graph construction to simplify the visual representation of large graphs. Power graphs serve as a novel representation that uses cliques, bicliques, and stars as primitives, thus reducing the number of original nodes and edges and facilitating the data interpretation. To be specific, we first developed a novel algorithm named Bitwise Clique Detection (BCD) to find the optimal maximum edge clique or biclique. The advantage of BCD over related work is the improvement of sets comparison using binary operations while computing the common neighbors between two nodes. We also obtained an improvement in edge reduction over previous work due to the fact that BCD maximizes the number of nodes that should be inserted in a power node. Furthermore, we demonstrated a simple orthogonal layout algorithm which emphasizes the minimization of the area occupied by graphic components, imposing an order on the diagram and reducing nodes and edges occlusion. As other contributions, we used the ordering of the power nodes found to reorder the adjacency matrix visualization and with the help of user interaction we used BCD output as a graph clustering technique. We conducted empirical evaluations for our clustering approach based on three machine learning datasets: zoo animals, cat brain connectome, and a dermatology diseases. In fact, by visualizing the results obtained from BCD clustering and comparing with popular clustering techniques we tested the ability of our technique to detect groups and find meaningful patterns, which validated the effectiveness of our approach. Finally, we use the main method to power graphs(biclique detection) to reduce black pixels of images of different sizes.

Limitations of the BCD algorithm are the need use to undirected graphs, for weighted directed graphs or graphs, so we planted a criterion for binarizing weighted matrices. With respect to orthogonal layout the fundamental limitation lies when drawing the edges, to draw an edge is necessary to explore within the graph, but in large graphs, the number of edges exceeds 1000 and run the Dijkstra algorithm for large graphs and run several times it does slow.

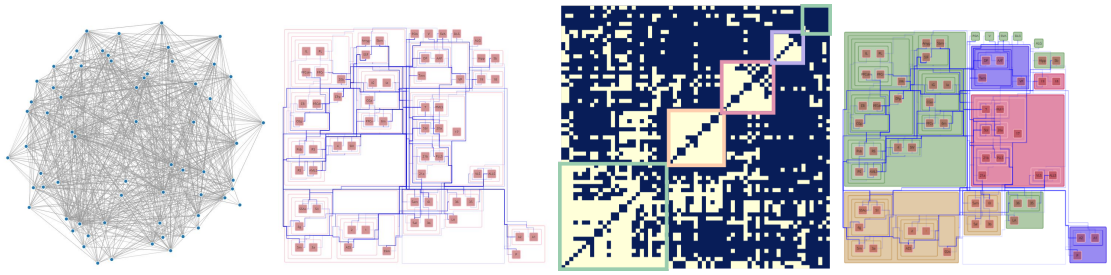
In the future, we plan to improve BCD to deal with weighted adjacency matrices without previous binarization to prevent loss of performance. We hope to expand the clustering technique using power graphs. Though we think that experts input on data analysis task is valuable, automatic tools for clustering analysis offer different insights. Additionally, we will explore interactive layouts for complex networks and cluster analysis.

APÊNDICE A — RESUMO EXPANDIDO EM PORTUGUÊS

Ao longo dos anos, os cientistas desenvolveram vários métodos para estudar redes do mundo real. Na neurociência, por exemplo, existe um grande interesse em encontrar informações sobre o comportamento das redes neurais (SPORNS; TONONI; KÖTTER, 2005; SPORNS, 2011; REUS; HEUVEL, 2013). Uma característica proeminente nesse campo é o estudo de conectomas, mas a complexidade dos seres vivos torna impossível estudar neurônios individuais na maioria dos casos. Para reduzir a complexidade, a análise é feita em regiões específicas do cérebro, como o conectoma humano (SPORNS; TONONI; KÖTTER, 2005) ou o conectoma de gatos (REUS; HEUVEL, 2013). Outras redes amplamente estudadas são as redes de proteínas (CHASSEY et al., 2008), redes sociais (BRANDES; WAGNER, 2004) e de citações (LESKOVEC; HORVITZ, 2014).

Os grafos são essenciais para muitas representações de dados. A análise visual de grafos é usualmente difícil devido ao tamanho, o que representa um desafio para sua visualização. Além de isso, seus algoritmos fundamentais são frequentemente classificados como NP-difícil. A análise de grafos de potência (AGP) é uma técnica proposta para simplificar grafos usando representações reduzidas, chamadas Grafos de Potência ou *Power Graphs*, que possuem representações especiais para subgrafos completos (cliques) e subgrafos bipartidos completos (bicliques). Os grafos de potência permitem uma redução considerável no número de arestas. A Figura A.1 ilustra um grafo denso (a) e sua representação correspondente no grafo de potência (b), que usa a terminologia especial de *nodo de potência* e *aresta de potência* para se referir aos seus vértices e arestas. Existem muitos grafos de potência possíveis associados a um determinado grafo, e um grafo de potência ideal é aquele que possui o menor número de arestas de potência, diretamente relacionado a o sucesso em encontrar bicliques e panelinhas. É provável que esse

Figura A.1: Processo de agrupamentos baseados na análise de grafos de potência aplicado no grafo representando o conectoma do grafo.



(a) Grafo Denso (b) Layout Ortogonal/Grafo de Potência (c) Matriz de Adjacência/Grafo de Potência (d) Agrupamentos encontrados com BCD

problema seja NP-difícil, pois encontrar a biclique máxima em um grafo mostrou-se NP-difícil (PEETERS, 2003). Portanto, os algoritmos de computação do grafo de potência baseiam-se em algoritmos de aproximação e heurísticas.

Uma abordagem hierárquica de agrupamento é usada em (ROYER et al., 2008) para computação de grafos de potência, mas é lenta para grafos densos. Dwyer et al. (DWYER et al., 2014) propuseram um algoritmo aprimorado usando uma estratégia gananciosa e que grafos de potência permitidos para a visualização de grafos densos. No entanto, seus resultados foram limitados a pequenos grafos (até 100 vértices). O objetivo desta dissertação é escalar a análise de grafos de potência para grafos maiores.

Neste trabalho, propusemos um novo algoritmo para construção de grafos de potência para simplificar a representação visual de grandes grafos. Os grafos de potência servem como uma representação nova que utiliza cliques, bicliques e estrelas como primitivas, reduzindo assim o número de nós e arestas originais, assim facilitando a interpretação dos dados. Primeiro desenvolvemos um novo algoritmo chamado *Bitwise Clique Detection* (BCD) para encontrar a biclique máxima ou clique ideal de borda máxima. A vantagem do BCD sobre o trabalho relacionado é a melhoria da comparação de conjuntos usando operações binárias enquanto computa os vizinhos comuns entre dois nós. Também obtivemos uma melhoria na redução de borda em relação ao trabalho anterior devido ao fato de o BCD maximizar o número de nós que devem ser inseridos em um nó de energia. Além disso, demonstramos um algoritmo de layout ortogonal simples que enfatiza a minimização da área ocupada por componentes grafos, impondo uma ordem ao diagrama e reduzindo a oclusão de nós e arestas. Como outras contribuições, usamos a ordem dos nós de energia encontrados para reordenar a visualização da matriz de adjacência e, com a ajuda da interação do usuário, usamos a saída BCD como uma técnica de agrupamento de grafos. Realizamos avaliações empíricas para nossa abordagem de agrupamento com base em três conjuntos de dados de aprendizado de máquina: animais do zoológico, conectoma do cérebro de gato e doenças dermatológicas. De fato, visualizando os resultados obtidos com o agrupamento de BCD e comparando com as técnicas populares de agrupamento, testamos a capacidade de nossa técnica em detectar grupos e encontrar padrões significativos, que validavam a eficácia de nossa abordagem. Finalmente, usamos o método principal para gerar grafos (detecção de biclique) para reduzir pixels pretos de imagens de tamanhos diferentes.

Em resumo, as principais contribuições apresentadas neste trabalho são:

- *Binary Clique Detection* (BCD), um método PGA que usa uma representação de

matriz bit a bit para melhorar a detecção de cliques e bicliques, e levar a uma aceleração de uma ordem de magnitude para métodos anteriores,

- Um layout ortogonal especialmente projetado para grafos de potência,
- Um algoritmo de grafo de potência para agrupar grafos densos que combinados com os layouts ortogonais permitem inspecionar os resultados do algoritmo de agrupamento, bem como a conectividade entre os nós em cada agrupamento.

As limitações do algoritmo BCD incluem grafos não direcionados, grafos direcionados ponderados ou grafos. Portanto, propomos um critério para a binarização de matrizes ponderadas. No que diz respeito ao layout ortogonal, a limitação fundamental está ao desenhar as arestas, é necessário explorar uma aresta para explorar dentro do grafo, mas em grafos grandes, o número de arestas excede 1000 e execute o algoritmo Dijkstra para grafos grandes e execute várias vezes desacelera.

No futuro, planejamos melhorar o algoritmo BCD para lidar com matrizes de adjacência ponderadas sem etapas de binarização anteriores, para evitar perda de desempenho. Esperamos expandir a técnica de agrupamento usando grafos de potência. Embora pensemos que a contribuição dos especialistas na tarefa de análise de dados seja valiosa, as ferramentas automáticas para análise de agrupamentos oferecem diferentes perspectivas. Além disso, exploraremos layouts interativos para redes complexas e análise de agrupamentos.

APÊNDICE — REFERÊNCIAS

BEAUXIS-AUSSALET, E.; HARDMAN, L. **Simplifying the Visualization of Confusion Matrix (Poster)**. [S.l.]: BNAIC, 2014.

BEAUXIS-AUSSALET, E.; HARDMAN, L. **Visualization of Confusion Matrix for Non-Expert Users (Poster)**. [S.l.]: IEEE, 2014.

BECK MICHAEL BURCH, S. D. F.; WEISKOPF, D. A taxonomy and survey of dynamic graph visualization. **Computer Graphics**, uni Stuttgart, 2015.

BONDY, J. A.; MURTY, U. S. R. **Graph theory with applications**. [S.l.]: Macmillan London, 1976.

BOSTOCK, M.; OGIEVETSKY, V.; HEER, J. D3: Data-driven documents. **IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)**, 2011. Available from Internet: <http://vis.stanford.edu/papers/d3>.

BRANDES, U.; WAGNER, D. Graph drawing software. In: ____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. chp. Analysis and Visualization of Social Networks, p. 321–340. ISBN 978-3-642-18638-7.

CHASSEY, B. D. et al. Hepatitis c virus infection protein network. **Molecular systems biology**, EMBO Press, v. 4, n. 1, p. 230, 2008.

COLIZZA, V. et al. Detecting rich-club ordering in complex networks. **Nature physics**, Nature Publishing Group, v. 2, n. 2, p. 110–115, 2006.

DAMINELLI, S. et al. Drug repositioning through incomplete bi-cliques in an integrated drug–target–disease network. **Integrative Biology**, Royal Society of Chemistry, v. 4, n. 7, p. 778–788, 2012.

DUDAS, P. M.; JONGH, M. d.; BRUSILOVSKY, P. A semi-supervised approach to visualizing and manipulating overlapping communities. In: **Information Visualisation (IV), 2013 17th International Conference**. [S.l.: s.n.], 2013. p. 180–185. ISSN 1550-6037.

DWYER, T. et al. Improved optimal and approximate power graph compression for clearer visualisation of dense graphs. In: IEEE. **Pacific Visualization Symposium (PacificVis), 2014 IEEE**. [S.l.], 2014. p. 105–112.

DWYER, T. et al. Edge compression techniques for visualization of dense directed graphs. **Visualization and Computer Graphics, IEEE Transactions on**, IEEE, v. 19, n. 12, p. 2596–2605, 2013.

EIGLSPERGER, M.; FÖSSMEIER, U.; KAUFMANN, M. Orthogonal graph drawing with constraints. In: CITESEER. **In Proc. 11th ACM-SIAM Symposium on Discrete Algorithms**. [S.l.], 2000.

EISEN, M. B. et al. Cluster analysis and display of genome-wide expression patterns. **Proceedings of the National Academy of Sciences**, National Acad Sciences, v. 95, n. 25, p. 14863–14868, 1998.

FALKOWSKI, B. J. Lossless binary image compression using logic functions and spectra. **Computers & Electrical Engineering**, Elsevier, v. 30, n. 1, p. 17–43, 2004.

GAGNEUR, J. et al. Modular decomposition of protein-protein interaction networks. **Genome biology**, BioMed Central Ltd, v. 5, n. 8, p. R57, 2004.

GUALDRON ROBSON L. F. CORDEIRO, J. F. R. J. H. Structmatrix: large-scale visualization of graphs by means of structure detection and dense matrices. **Data Mining Workshops**, icmc USP, 2015.

HEUVEL, M. P. V. D.; POL, H. E. H. Exploring the brain network: a review on resting-state fmri functional connectivity. **European Neuropsychopharmacology**, Elsevier, v. 20, n. 8, p. 519–534, 2010.

HEUVEL, M. P. van den; SPORNS, O. Network hubs in the human brain. **Trends in cognitive sciences**, Elsevier, v. 17, n. 12, p. 683–696, 2013.

HU, Y.; SHI, L. Visualizing large graphs. **Wiley Interdisciplinary Reviews: Computational Statistics**, John Wiley & Sons, Inc., v. 7, n. 2, p. 115–136, 2015. ISSN 1939-0068.

JACCARD, P. **Etude comparative de la distribution florale dans une portion des Alpes et du Jura**. [S.l.]: Impr. Corbaz, 1901.

JANKUN-KELLY, T. J. et al. Multivariate network visualization: Dagstuhl seminar #13201, dagstuhl castle, germany, may 12-17, 2013, revised discussions. In: ____. Cham: Springer International Publishing, 2014. chp. Scalability Considerations for Multivariate Graph Visualization, p. 207–235. ISBN 978-3-319-06793-3.

KIEFFER, S. et al. Hola: Human-like orthogonal network layout. **IEEE Transactions on Visualization and Computer Graphics**, v. 22, n. 1, p. 349–358, Jan 2016. ISSN 1077-2626.

LESKOVEC, J.; HORVITZ, E. Geospatial structure of a planetary-scale social network. **Computational Social Systems, IEEE Transactions on**, IEEE, v. 1, n. 3, p. 156–163, 2014.

LEVINSON, M. H. Linked: The new science of networks. **ETC.: A Review of General Semantics**, Institute of General Semantics, v. 61, n. 1, p. 170–171, 2004.

LICHMAN, M. **UCI Machine Learning Repository**. 2013. Available from Internet: <http://archive.ics.uci.edu/ml>.

MARTI, G. et al. Hcmapper: An interactive visualization tool to compare partition-based flat clustering extracted from pairs of dendrograms. **CoRR**, abs/1507.08137, 2015. Available from Internet: <http://arxiv.org/abs/1507.08137>.

MOHAMED, S. A.; FAHMY, M. M. Binary image compression using efficient partitioning into rectangular regions. **Communications, IEEE Transactions on**, IEEE, v. 43, n. 5, p. 1888–1893, 1995.

MRžEK, M.; BLAžIč, B. J. Fast network communities visualization on massively parallel gpu architecture. In: **Information Communication Technology Electronics Microelectronics (MIPRO), 2013 36th International Convention on**. [S.l.: s.n.], 2013. p. 269–274.

PEETERS, R. The maximum edge biclique problem is np-complete. **Discrete Applied Mathematics**, Elsevier, v. 131, n. 3, p. 651–654, 2003.

PHIPPS, J. Dendrogram topology. **Systematic Biology**, Oxford University Press, v. 20, n. 3, p. 306–308, 1971.

REUS, M. A. de; HEUVEL, M. P. van den. Rich club organization and intermodule communication in the cat connectome. **The Journal of Neuroscience**, Soc Neuroscience, v. 33, n. 32, p. 12929–12939, 2013.

ROYER, L. et al. Unraveling protein networks with power graph analysis. 2008.

RUNPENG, L.; JUN, H.; XIAOFAN, W. Vcd: A network visualization tool based on community detection. In: **Control, Automation and Systems (ICCAS), 2012 12th International Conference on**. [S.l.: s.n.], 2012. p. 1221–1226.

SPORNS, O. **Networks of the Brain**. [S.l.]: MIT press, 2011.

SPORNS, O.; TONONI, G.; KÖTTER, R. The human connectome: a structural description of the human brain. **PLoS Comput Biol**, v. 1, n. 4, p. e42, 2005.

WATTS, D. J.; STROGATZ, S. H. Collective dynamics of 'small-world'networks. **nature**, Nature Publishing Group, v. 393, n. 6684, p. 440–442, 1998.

WONG, P. C. et al. Visual matrix clustering of social networks. **IEEE Computer Graphics and Applications**, v. 33, n. 4, p. 88–96, July 2013. ISSN 0272-1716.

WU, H.-M.; TIEN, Y.-J.; CHEN, C. houh. Gap: A graphical environment for matrix visualization and cluster analysis. **Computational Statistics and Data Analysis**, v. 54, n. 3, p. 767 – 778, 2010. ISSN 0167-9473. Second Special Issue on Statistical Algorithms and Software. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0167947308004349>.

ZHANG, Y.; PARTHASARATHY, S. Extracting analyzing and visualizing triangle k-core motifs within networks. In: **Data Engineering (ICDE), 2012 IEEE 28th International Conference on**. [S.l.: s.n.], 2012. p. 1049–1060. ISSN 1063-6382.