José Antônio Pellizzaro

Estudo de Diferentes Algoritmos Visando o Bandiwth Minimization

Porto Alegre

2016

José Antônio Pellizzaro

Estudo de Diferentes Algoritmos Visando o Bandiwth Minimization

Trabalho de conclusão de curso como um dos pré-requisitos para a obtenção do grau de Bacharel em Física, apresentado ao Instituto de Física da Universidade Federal do Rio Grande do Sul

Universidade Federal do Rio Grande do Sul - UFRGS

Instituto de Física

Curso de Física: Pesquisa Básica

Orientador: Daniel Gamermann

Porto Alegre

2016

Resumo

A quantidade de informações disponiveis sobre sistemas complexos como a internet, redes sociais e biológicas nunca foi tão grande. Por isso, desenvolver as ferramentas computacionais necessárias para analisar essas informações é uma tarefa que merece muita atenção. Nesse trabalho, estamos interessados em um problema em particular: encontrar as comunidades de uma rede metabólica ou de interação proteína-proteína (PPI de *protein-protein interaction*). Devido a complexidade computacional do problema nós desenvolvemos dois algoritmos heurísticos e os comparamos com dois algoritmos tradicionais: Kruskal e PageRank. Muito embora os ordenamentos obtidos pelos nossos algoritmos pareçam aproximar os nós de uma maneira mais efetiva, ao compararmos nossos resultados com os termos GO (de *Gene Ontology*, que relacionam as proteínas com suas funções biológicas) associados não encontramos forte correlação.

Palavras-chave: Redes Metabólicas, Redes PPI, comunidades, PageRank,Kruskal, termos GO.

Abstract

The amount of available data on complex systems such as the internet, social and biological networks has never been bigger. As such, developing the necessary computational tools to handle this data is a task of great importance. In this work we are interested in one particular problem: finding the community structure of a metabolic or PPI network. Due to it's computational complexity we propose two heuristic algorithms to tackle this problem and compare them to two traditional ones, namely Kruskal and PageRank. Even though our algorithms seem to be better at clustering the networks, when we compared our results with those of the GO terms (from Gene Ontology, they relate the proteins in the network to their biological functions) no strong correlation was found.

Keywords: Metabolic Networks, PPI Networks, comunity, PageRank, Kruskal, GO terms.

Sumário

Sumário 5

Introdução 7

1 PROPRIEDADES 9
1.1 Comunidades 11
1.2 Obtenção das redes 12

2 ALGORITMOS 13
2.1 Complexidade computacional 13
2.2 PageRank 14
2.3 Kruskal 16
2.4 Monte Carlo 16
2.4.1 Modelo 17
2.4.1.1 Forças 17
2.4.2 Algoritmo 19
2.4.3 Blocos 20
2.4.4 Ordenamento Inicial 23
2.5 Clustering por similaridade 23

3 RESULTADOS 27
3.1 Redes PPI 29
3.1.1 *Candidatus Hodgkinia cicadicola Dsem* 29
3.1.2 *Mesoplasma florum L1* 32
3.2 Redes Metabólicas 36
3.2.1 *Synechocystis sp PCC. 6803* 36
3.2.2 *Mycoplasma genitalium G37* 39

4 CONCLUSÃO 43

REFERÊNCIAS 45

Introdução

A teoria dos grafos teve sua origem na solução, por Leonard Euler, do problema das sete pontes de Königsberg. A cidade de Königsberg era cercada pelo rio Pregel, de modo que o deslocamento entre as suas diferentes partes era feito através de sete pontes. A questão, que motivou Euler, diz respeito a essas pontes e é formulada da seguinte maneira: é possível atravessar todas as pontes sem nunca passar pela mesma ponte duas vezes?

Euler percebeu que não se tratava de um problema de geometria, ou da topografia da cidade, mas de como as diferentes áreas da cidade concectavam-se entre si. Por isso, representou cada área como um ponto - formalmente chamado de vértice e, onde havia pontes, desenhou linhas conectando esses pontos - que chamamos de ligações. Ao análisar as propriedades dessa figura, Euler observou que era impossível obter uma trajetória que atravessase todas as pontes apenas uma vez. Esse tipo de representação, que usa vértices e ligações, é o que chamamos de grafo $G(V, L)$.

Quase 300 anos após o artigo inicial de Euler, usamos grafos para representar as redes sociais [1], a internet [2], redes biológicas [3] [4], de transmissão de energia, de parceiros sexuais, etc. Muito embora a estrutura matemática que descreva esses sistemas seja a mesma, o seu significado físico varia de sistema para sistema. Por causa disso, a maneira como se definem os nós e as ligações é de suma importância, pois limitam toda a análise posterior do problema (nesse texto utilizamos nós, conexões e redes como sinônimos para vertíces, ligações e grafos, respectivamente) . Por exemplo, nas redes de parceiros sexuais , os nós simbolizam pessoas e as ligações representam relações íntimas, isso é feito pois, com essa escolha é possível estudarmos a propagação de DSTs [5].

Nesse trabalho analisaremos dois tipos de redes biológicas: redes metabólicas e redes de interação de proteínas ou PPI (para *protein-protein interaction*). Nas redes PPI cada vértice representa uma proteína e uma ligação entre dois vértices nos diz que essas proteínas interagem entre si em determinado processo no organismo. Já nas redes metabólicas cada nó simboliza um metabólito e uma ligação entre dois metabólitos nos diz que eles fazem parte de uma mesma reação química como um par produto-reagente.

A propriedade dessas redes que nos interessa especialmente é a presença de *clusters* ou comunidades [1]. Um *cluster* é um subgrupo de nós que contêm muitas conexões entre si, mas que são fracamente conectados com o restante da rede. Nas redes biológicas que estudamos, moléculas ou proteínas que fazem parte de uma mesma comunidade tem mais chances de participar de um mesmo ciclo ou função no organismo [4] [3]. Ser capaz de encontrar as comunidades também é relevante em outros tipos de redes. Nas redes sociais, uma comunidade pode simbolizar relações familiares ou de trabalho, em redes de citações,

podemos encontrar grupos que trabalham com o mesmo assunto, ou ainda, podemos encontrar páginas sobre um mesmo tópico na internet.

No entanto, devido a sua complexidade computacional (por exemplo [6]), não existe um método definitivo para solucionar o problema de encontrar os *clusters* de uma rede. Nesse trabalho propomos novos algoritmos heurtísticos e os comparamos com métodos tradicionais: Kruskal [7] e PageRank [2].

1 Propriedades

Um grafo ou uma rede $G(V, L)$, como já vimos, é formado por um conjunto de vértices V (nós ou pontos) e de conexões L entre eles (figura 1). Para podermos estudar as propriedades associadas com essas estruturas é necessario uma forma de descrevê-las matematicamente. Para isso, utilizamos a matriz de adjacência, que representa as ligações entre os nós da rede. Uma rede com N nós é transcrita numa matriz quadrada NxN onde cada elemento A_{ij} é definido por:

$$A = \begin{cases} A_{ij} = 1, \text{ se há uma conexão entre os nós i e j.} \\ A_{ij} = 0, \text{ caso contrário} \end{cases}$$

Como as redes metabólicas e PPI são não-direcionadas a matriz A é simétrica, ou seja, se o nó i é ligado ao nó j isso implica que o nó j é conectado ao nó i. Consideremos o grafo representado na figura 1, a matriz de ajdacência respectiva será:

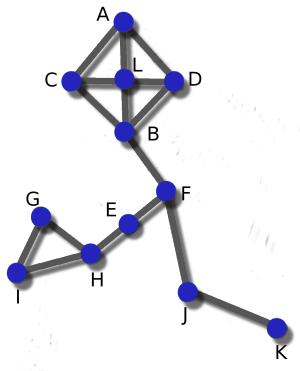


Figura 1 – Um exemplo de grafo com 12 vértices (representados pelas letras de A a L) e 16 ligações.

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.1}$$

Para obter o número de conexões (e de vizinhos) de um nó na rede, basta fazermos uma soma sob as linhas ou colunas da matriz A (equação 1.2). A essa característica se dá o nome de grau do nó (k_i). Nós com muitas conexões e, portanto, com k_i elevado, formam um grupo especial e são chamados de *hubs*.

$$k_i = \sum_{j=1}^{N} A_{ij} \tag{1.2}$$

Como o grau varia de nó para nó, dizemos que é uma propriedade local da rede. Já características que consideram a rede como um todo são chamadas de propriedades globais. Uma dessas propriedades é o coeficiente de agrupamento global (*global clustering coeficient*) denotado por C, ele nos diz que dois nós que compartilham um mesmo vizinho tem uma probabilidade fixa de serem conectados entre si. Esse conceito pode ser explicado intuitivamente se usarmos redes sociais. Basicamente, estamos afirmando que dois de seus amigos tem maior probabilidade de se conhecerem do que conhecer uma pessoa aleatória da população. Calculamos C através da seguinte expressão:

$$C = \frac{3 \times \text{ numero de triangulos no grafo}}{\text{numero de trios conectados de vertices}} \tag{1.3}$$

Onde um trio conectado é um conjunto de vértices ABC, onde A se conecta com B e B se conecta com C (em outras palavras, um trio é um caminho de tamanho 2) de A para C. Cada triângulo conta como 3 trios, por esse motivo temos o fator 3 no numerador. Na figura 1, por exemplo, os nós HIG foram um triângulo, que é o conjunto dos trios HIG, IGH e GHI.

Existe também o coeficiente de agrupamento local C_i que, por sua vez, representa a probabilidade de que os vizinhos de um nó i sejam conectados. Ele é dado por:

$$C_i = \frac{\text{numero de triangulos entre os vizinhos do nó i}}{\text{numero de triangulos possiveis entre seus vizinhos}} \tag{1.4}$$

O C_i de um nó é, de fato, uma estimativa da densidade local da rede. Analisando, novamente, o triângulo GHI da figura 1, temos que $C_G = C_I = 1$ e $C_H = \frac{1}{3}$ enquanto $C_E = 0$.

Outra propriedade é o *topological overlap*. Ele é usado para identificar como um par de nós se relaciona com todos os outros na rede. O topological overlap To_{ij} é uma medida da semelhança entre os vizinhos de um par de nós i e j que é definida como:

$$To_{ij} = \frac{V_{ij} + \begin{cases} 1, \text{se os nós i e j são conectados} \\ 0, \text{caso contrário} \end{cases}}{min(k_i, k_j)} \tag{1.5}$$

Na qual V_{ij} são os vizinhos em comum entre os dois nós e $min(k_i, k_j)$ é o menor grau entre os nós i e j. O *topological overlap* varia de 0 a 1, sendo 0 se os nós não forem conectados e não tiverem nenhum vizinho em comum, e 1, se os nós i e j forem conectados e se o nó de maior grau contiver todos os vizinhos do nó de menor grau. Essa informação é de extrema relevância, pois nós num mesmo agrupamento devem compartilhar muitas conexões entre si, o que equivale a dizer que o *topological overlap* entre os membros de um *cluster* deve ser alto.

1.1 Comunidades

Antes de nos voltarmos para o problema de encontrar comunidades numa rede, devemos nos focar em buscar uma definição apropriada para uma comunidade. Intuitivamente, uma comunidade é um subconjunto de nós densamente conectados, mas que tem poucas conexões com outras comunidades. Aqui surge o primeiro desafio, pois não existe um consenso de como quantificar esse conceito.

No entanto, existem alguns aspectos gerais que são comuns a maioria das definições. Consideremos uma comunidade $g(n, l)$ que é um subgrafo de $G(N, L)$, dentro dessa comunidade existem m conexões entre seus membros de modo que podemos definir um grau "interno"da comunidade como $k_{in} = m$, por outro lado existem p conexões entre nós pertencentes a comunidade e nós externos com as quais definimos o grau "externo"da comunidade $k_{out} = p$. Para que $g(n, l)$ seja uma comunidade é necessário que $k_{in} >> k_{out}$ e que $g(n, l)$ seja conexo, isto é, podemos passar por todos os nós do agrupamento percorrendo um caminho pelas ligações do subgrafo.

1.2 Obtenção das redes

As redes metabólicas são geradas a partir da base de dados do KEGG (*Kyoto Encyclopedia of Genes and Genomes*) [8]. No KEGG estão disponíveis informações sobre os genes, as vias metabólicas, as enzimas, as reações e os componentes das reações presentes nos organismos em questão. A confecção das redes é feita por um processo automatizado descrito em detalhe em [9]. É importante salientar que ignoramos o sentido das reações químicas nesse trabalho, de modo que a rede metabólica gerada é não-direcinada.

As redes PPI, por sua vez, são geradas a partir do *String* [10]. Nesse Database são atribuídos valores de confiabilidade (entre 0 e 1000) para cada interação entre duas proteínas. Esses valores levam em consideração dados experimentais, informações da literatura, informações prévias de outras bases de dados, entre outros. Construímos as redes de maneira que duas proteínas são conectadas se o valor de confiabilidade entre elas for maior ou igual a 900, isto é se tivermos ao menos 90% de confiabilidade de que as proteínas interagem entre si.

2 Algoritmos

Nesse trabalho desenvolvemos dois algoritmos heurísticos para a identificação das comunidades de uma rede. Esse problema pode ser considerado como uma variação do *Matrix Bandwidth minimization problem* (MBMP) [11] , nessa formulação o que se busca é a permutação das colunas e linhas de uma matriz que deixem os elementos não nulos desta o mais próximos possíveis da diagonal principal, como veremos a seguir, buscamos aproximar os elementos não-nulos da matriz do *topological overlap* com nossos algoritmos. Comparamos nossos resultados com os obitidos pelo PageRank [2] e o Kruskal [7] que são usados tradicionalmente. Para entender porque são necessários algoritmos heurísticos, começaremos tratando da complexidade computacional do problema.

2.1 Complexidade computacional

A complexidade computacional de um problema está relacionada com a quantidade de passos computacionais e de memória necessários para resolvê-lo. Problemas com complexidade polinomial formam a classe **P**, nessa classe o tempo de execução dos algoritmos é limitado por uma função polinomial do tamanho do problema. Note que o tempo ao qual nos referimos não é medido em minutos e segundos mas em número de passos executados. Por exemplo qualquer algoritmo com tempo de execução dado por $O(N^k)$ está contido na classe **P**, onde N é o tamanho da entrada do algoritmo (por exemplo N pode ser o número de ligações ou de nós de uma rede).

Existem outros problemas que não tem um algoritmo polinomial para resolvê-los, mas cujas soluções, se conhecidas, podem ser verificadas em tempo polinomial. A essa classe se dá o nome de **NP**. No pior dos casos, é necessário usar métodos de força bruta para encontrar a soluções para estes problemas, com esse método o tempo de execução aumenta mais rápido que qualquer função polinomial do tamamho do sistema (para exemplificar: o número possível de partições de uma rede cresce mais rápido que uma exponencial). Na verdade, esses problemas talvez nem tenham solução em tempo polinomial, a questão de $P = NP$, isto é, se para qualquer problema em **NP** existe uma solução em **P**, ainda está em aberto.

Dentro da classe **NP** existem outras classificações. Um problema é **NP**-*hard* se a sua solução pode ser adaptada para qualquer outro problema em **NP**, no entanto, um problema **NP**-*hard* não precisa estar contido em **NP**. Se ele estiver em **NP** temos o que chamamos de **NP**-*complete*. A maioria dos problemas que envolvem o *clustering* de redes pertencem a classe **NP** (por exemplo o problema de partição dos grafos, de encontrar a divisão que maximize a modularidade [6] são **NP**-*hard*, já o MBMP é **NP**-complete) [12],

isso torna inviável o uso de algoritmos que deem a solução exata do problema, pois estes, devido ao seu grande tempo de execução, só podem ser aplicados em sistemas muito pequenos. Por outro lado, o uso de algoritmos heurísticos é muito comum, pois é possível obter uma solução aproximada do problema com uma perda considerável de complexidade.

2.2 PageRank

O PageRank era um algoritmo utilizado pelo google para ordenar os resultados de uma busca em seu site. Começaremos falando sobre como esse algoritmo foi desenvolvido e então consideraremos as adaptações necessárias para aplicá-lo em redes metabólicas e redes PPI.

O google representa a internet como um grande grafo, onde cada vértice representa um site e cada ligação um link entre duas páginas da web. Como sempre, representamos esse grafo através da sua matriz de adjacência A onde um elemento a_{ij} será 1 se houver um link da página P_j para a página P_i e 0 caso contrário. Nesse caso, estamos falando de um grafo direcionado pois A conter um link para B não implica que B contenha um link para A. Por causa disso uma soma sobre a coluna de A_j nos dá os links que saem dessa página, já uma soma sob a linha de A_j nos dá o número de links que chegam em A_j.

O que queremos é atribuir um valor de relevância x_j para cada uma das páginas P_j que fazem parte do grafo. Para isso devemos considerar dois fatores distintos. O primeiro deles é que páginas mais citadas, isto é, com mais links que levam a elas devem ter uma relevância maior que páginas menos citadas. No entanto, essa definição sozinha nos deixa com um problema: e se tivermos uma página com pucas sitações mas essas sitações forem de páginas muito relevantes?

O jeito encontrado pelo google para resolver esse problema é tornar a relevância de uma página P_j proporcional a soma das relevâncias das páginas que tem links para P_j. Supondo um sistema onde P_2 é citada por P_1,P_{n-1}, P_n, que P_n cita P_1, P_2, P_{n-1} e é citada por P_1, que P_{n-1} faz referência à P_2 e recebe links de P_2 e P_n. As relevâncias x_1,x_2, x_{n-1} e x_n obedecem as seguintes relações:

$$x_1 = K(x_n) \tag{2.1}$$

$$x_2 = K(x_1 + x_{n-1} + x_n) \tag{2.2}$$

$$\vdots$$

$$x_{n-1} = K(x_2 + x_n) \tag{2.3}$$

$$x_n = K(x_1) \tag{2.4}$$

Onde K é uma constante de proporcionalidade. O que temos aqui é um sistema de equações lineares e podemos, portanto, reescrevê-lo em notação matricial, da forma:

$$\begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_{n-1} \\ x_n \end{pmatrix} = K \begin{pmatrix} 0 & 0 & . & . & . & 0 & 1 \\ 1 & 0 & . & . & . & 1 & 1 \\ . & . & & & & . & . \\ . & . & & & & . & . \\ . & . & & & & . & . \\ 0 & 1 & . & . & . & 0 & 1 \\ 1 & 0 & . & . & . & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_{n-1} \\ x_n \end{pmatrix} \tag{2.5}$$

A matriz $N \times N$ no lado direito da igualdade é justamente a matriz de adjacência A. Podemos chamar os valores $x_1, x_2, .., x_n$ de vetor relevância x e a constante de proporcionalidade de λ dessa forma o problema se traduz em:

$$AX = \lambda X \tag{2.6}$$

Que nada mais é que um problema de autovalores e autovetores. Portanto, vemos claramente que X é um autovetor da matriz A. Além disso gostaríamos que todos as elementos desse vetor (as relevâncias x_j) fossem positivas $x >= 0$. Como condição final precisamos que esse vetor X seja único.

Nesse ponto o teorema de Perron-Frobenius se faz relevante. Ele diz que:

Teorema (Frobenius, 1908-1912) 1 *Se A é uma matriz quadrada, não-negativa e irredutível, temos que:*

1. *Existe um autovalor (simples) $\lambda > 0$ tal que $Av = \lambda v$, onde o autovetor correspondente é $v > 0$. Além disso, $\lambda >= |\mu|$ para cada outro autovalor de A.*
2. *Qualquer autovetor $>= 0$ é um múltipo de v*
3. *Se existem k outros autovalores de módulo máximo, então eles são solução da equação $x^k - \lambda^k = 0$*

Os elementos de A são não negativos por definição, se A for irredutível o nosso problema estará resolvido, pois existe um autovetor único e não-negativo associado ao autovalor de maior módulo. Nesse ponto, nos voltamos as redes metabólicas e PPI.

As redes metabólicas e PPI, ao contrário da internet, são não-direcionadas, ou seja a matriz de adjacência A destas redes também é simétrica. Para que um grafo não-direcionado seja considerada irredútivel, ele deve ser fracamente conectado. Um grafo fracamente conectado contem um caminho que ligue qualquer par de nós i e j no grafo. Como consideramos a maior componente conexa da rede, temos que A é irredutível e

portanto resolvemos o problema, pois basta encontrar o autovalor de maior módulo e calcularmos o autovetor v associado, pois as suas componentes nos dão a relevância de cada nó.

2.3 Kruskal

O Kruskal é um algoritmo que encontra a árvore geradora mínima de um grafo conexo e com pesos. Como sempre utilizamos a maior componente conexa do grafo, não precisamos nos preocupar com a primeira parte da definição pois ela é evidente. Porém, as redes que utilizamos não tem um peso atribuído a priori a suas ligações. Antes de definirmos o critério que utilizaremos, é conveniente citar os passos do algoritmo.

- Inicialmente separa-se o grafo $G(V, L)$ em um cojunto de ligações $S(L)$ onde cada ligação tem um determinado peso, e um conjunto de vértices $F(V)$. Nesse ponto cada vértice é considerado uma árvore, por isso F é usualmente chamado de floresta.
- Remove-se a ligação com menor peso de $S(L)$. Se ela conectar duas árvores diferentes em F as duas são unidas formando uma única árvore e a ligação k é adicionada a F. Caso contrário a ligação é descartada.
- Os passos são repetidos enquanto houverem ligações em S.
- Então retorna F, que é a floresta que contem todos os vértices com a menor soma possível dos pesos das ligações.

O Kruskal é um algoritmo que faz um *hierarchical clustering*, isto é, atribui uma hierarquia aos vértices do grafo. Mais especificamente, ele faz um *clustering* aglomerativo, pois a partir de um estado inicial onde todos os vértices estão separados, cada um é uma árvore (ou comunidade) individual, eles vão sendo unidos iterativamente até pertencerem a uma única árvore.

Nesse tipo de algoritmo é necessário definir uma medida de semelhança entre os vértices, que serve de base para fazer a união entre eles. No caso do Kruskal, representamos essa semelhança através do peso das ligações.

Novamente utilizamos o *topological overlap* entre os nós como peso das ligações da rede, com essa definição, queremos a árvore que maximiza o peso das ligações. Para isso basta removermos a ligação com maior peso de $S(L)$ a cada iteração do algoritmo.

2.4 Monte Carlo

O primeiro algoritmo desenvolvido faz um processo de Monte Carlo na rede a fim de encontrar suas comunidades. Para isso, partimos das definições que seguem.

2.4.1 Modelo

Começamos dispondo os nós num espaço unidimencional discreto. Imagine que temos N "caixas"unidimencionais enfileiradas e que cada uma dessas caixas possa conter somente um nó. Então atribuimos um número inteiro $i \in [1, N]$ para cada um desses nós, de modo que possamos identificá-los idividualmente. Chamamos a lista desses números de ordenamento O da matriz A. Como distribuimos os nós nas caixas é o que determina o ordenamento inicial do problema, conforme discutimos na sessão (2.4.4).

2.4.1.1 Forças

O ponto central de nosso modelo é que existe uma força F_{ij} entre cada par i e j de nós no ordenamento. Nós podem ser atráidos ou repelidos pelos outros da seguinte forma:

$$\begin{cases} F_{ij} = -To_{ij}s_{ij}, \text{ se o topologial overlap entre os nós i e j é diferente de zero.} \\ \\ F_{ij} = \frac{10}{s_{ij}}, \text{ se o topological overlap é zero.} \end{cases}$$

Onde s_{ij} é a separação entre os nós no ordenamento (note que a separação dos nós no ordenamento não é necessariamente relacionada com o menor caminho entre os nós na rede) e To_{ij} é o *topological overlap* entre os nós. Quando a força é atrativa, temos um sistema massa-mola onde a constante elástica é representada pelo *topological overlap* . Já a força repulsiva é dada por uma constante e diminui com a distância entre os nós. Definimos a constante como 10 para que a força repulsiva seja significativa, pois o To_{ij} só assume valores entre 0 e 1. Testes feitos com outros valores não mostraram mudanças significativas.

O argumento por trás dessa definição é que nós com To_{ij} alto tem maior probabilidade de fazerem parte de um mesmo *cluster* e portanto de estarem próximos em um ordenamento ótimo. Por exemplo, podemos confeccionar a matriz (M) do *topological overlap* do grafo da figura 1, nessa matriz cada elemento representa o To_{ij} entre os nós. Repare que entre A e B temos $To_{ij} = 1$ mesmo que os nós não sejam conectados, pois todos os vizinhos de A (menor grau) são também vizinhos de B.

$$M = \begin{pmatrix} & A & B & C & D & E & F & G & H & I & J & K & L \\ A & 1 & 1 & 0.66 & 0.66 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ B & 1 & 1 & 0.66 & 0.66 & 0.5 & 0.33 & 0 & 0 & 0 & 0.5 & 0 & 0.75 \\ C & 0.66 & 0.66 & 1 & 1 & 0 & 0.33 & 0 & 0 & 0 & 0 & 0 & 1 \\ D & 0.66 & 0.66 & 1 & 1 & 0 & 0.33 & 0 & 0 & 0 & 0 & 0 & 1 \\ E & 0 & 0.5 & 0 & 0 & 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0 & 0 \\ F & 0 & 0.33 & 0.33 & 0.33 & 0.5 & 1 & 0 & 0.33 & 0 & 0.5 & 1 & 0.33 \\ G & 0 & 0 & 0 & 0 & 0.5 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ H & 0 & 0 & 0 & 0 & 0.5 & 0.33 & 1 & 1 & 1 & 0 & 0 & 0 \\ I & 0 & 0 & 0 & 0 & 0.5 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ J & 0 & 0.5 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 1 & 1 & 0 \\ K & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ L & 1 & 0.75 & 1 & 1 & 0 & 0.33 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.7)$$

Para facilitar a visualização, na figura 2-(a) o *topological overlap* entre os nós está representado em cada ligação (a largura da ligação também reflete o To_{ij}). Os nós A, B, C, D e L formariam uma comunidade, pois é um grupo conexo e com $k_{in} = 8$ e $k_{out} = 1$. Nesse subgrafo o menor *topological overlap* entre seus membros é $To_{ij} = 0.5$. Da mesma forma G, H, I representam outro agrupamento com $k_{in} = 3$ e $k_{out} = 1$. Conforme dito anteriormente, nós que não estejam conectados podem apresentar $To_{ij} \neq 0$, por isso ao representarmos o grafo como em 2-(a) estamos omitindo informações sobre M. Em 2-(b) temos o gráfico de M, que é a maneira usual de representá-la. Células brancas simbolizam $To_{ij} = 0$, células pretas $To_{ij} = 1$ e os diferentes tons de cinza simbolizam os valores intermediários.

Como queremos evidenciar as comunidades gostaríamos de que nos com *topological overlap* grande fiquem próximos no ordenamento, por isso consideramos que a força atrativa é proporcional ao To_{ij} e a separação dos nós.

Usando essas forças, podemos facilmente obter a energia potencial associada através da relação:

$$-\frac{dU_{ij}}{ds} = \overrightarrow{F}_{ij} \quad (2.8)$$

Basta realizarmos uma integração para obtermos, em cada caso:

$$\begin{cases} U_{ij} = \frac{1}{2} To_{ij} s_{ij}^2, \text{se } 0 < To_{ij} \leq 1. \\ U_{ij} = -10 ln|s|, \text{se } To_{ij} = 0. \end{cases}$$

Com essas definições, no entanto, surguem dois problemas. O primeiro deles é que vértices sem ligações tem um *topological overlap* infinito porque possuem $k_i = 0$, conforme equação (4). Tratamos esse problema removendo esses vértices do grafo, de fato, sempre

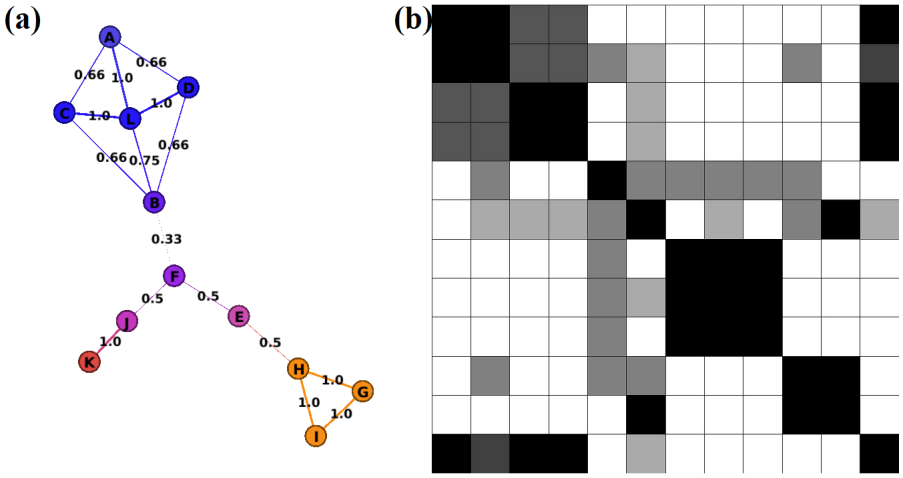


Figura 2 – (a): Representação do grafo da figura 1 e o *topological overlap* de suas ligações. Os conjuntos $ABCDL$ e GHI seriam comunidades desse grafo. (b): Matriz M do *topological overlap* desse mesmo grafo. Quanto mais escura a célula mais próximo de 1 o valor do To_{ij} entre os nós na linha e coluna.

utilizamos a maior componente conexa da rede, ou seja , de um grafo $G(V, L)$ retiramos um subgrafo $g(v, l)$ tal que g contenha o maior subconjunto de nós conectados por um caminho (um caminho é uma rota que percorre a rede atráves de suas ligações).

Além disso nós próximos das extremidades do ordenamento teriam uma assimetria na distribuição de vizinhos, por exemplo, como estamos num espaço unidimencional, o primeiro nó do ordenamento só teria vizinhos em uma direção com separação $s \in [1, N]$. Já o nó exatamente na metade do ordenamento teria vizinhos com $s \in [1, N/2]$. Para conciliar essas duas situações utilizamos condições periódicas de contorno, dessa forma garantimos que a separação máxima entre dois nós será sempre $N/2$ para N par e $N/2 + 1$ para N ímpar.

2.4.2 Algoritmo

Nosso algoritmo pode ser resumido nos seguintes passos:

- Atribuímos um ordenamento inicial para a rede $O(t = 0)$.
- Então selecionamos um grupo de três nós adjacentes nesse ordenamento e calculamos a força resultante sobre os três nós, que é dada por:

$$F_R = \sum_{j=1}^{N} F_{ij} \tag{2.9}$$

- Reordenamos os nós de acordo com a força resultante. Como temos um sistema unidimencional, nós à direita com força resultante para a esquerda são substituídos

por nós à esquerda com força resultante para a direita. Obtemos então um novo ordenamento $O(t + dt)$.

- Após n passos temporais o algoritmo retorna o novo ordenamento da rede.

Como consequência da execução do algoritmo, a energia potencial U associada ao sistema deve ser minimizada. Na figura 3 temos o gráfico de U pelo número de *Monte Carlo steps* da rede metabólica da *Synechocystis sp PCC. 6803* uma rede com 944 nós.

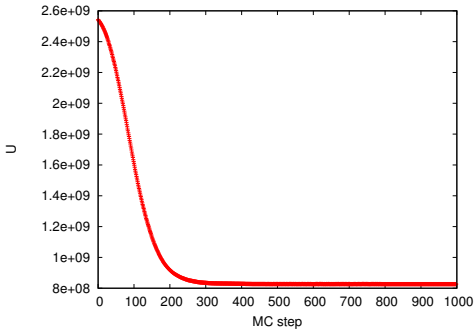


Figura 3 – Gráfico da energia potencial U em função do número de *Monte Carlo Steps* (MCS) para a rede metabólica da *Synechocystis sp PCC. 6803* que contêm 944 nós.

Observamos que o sistema atinge um mínimo de energia por volta de 300 MCS. No entanto, há a possibilidade de que este valor de energia represente um mínimo local, isso ocorre porque cada iteração do algoritmo diminui a energia do sistema ($\Delta U \leq 0$) de modo que ao encontrarmos um estado meta-estável não temos como fazer uma variação $\Delta U > 0$ na energia para sair desse estado. Nesse sentido, não fazemos um *simulated annealing* no sistema (como descrito em [13] e feito em [3] e [4]), para fazer alterações que possam aumentar a energia potencial do sistema fazemos uma variação no algoritmo.

2.4.3 Blocos

Inicialmente, consideramos a matriz do *topological overlap* da *Synechocystis sp PCC. 6803* depois de 1000 MCS (figura 4), fica claro que existem nós que compartilham um To_{ij} grande entre si, mas que ainda estão separados no ordenamento (visualmente, nos referimos as células escuras longe da diagonal principal). Isso ocorre porque a cada passo temporal o algoritmo considera um nó e seus vizinhos imediatos no ordenamento, se dois nós com um To_{ij} grande entre si ficarem muito distantes no ordenamento inicial é possível que o algoritmo não consiga aproximá-los. A maneira que encontramos para tratar dessa situação também provoca as variações $\Delta U > 0$ que gostaríamos na energia potencial.

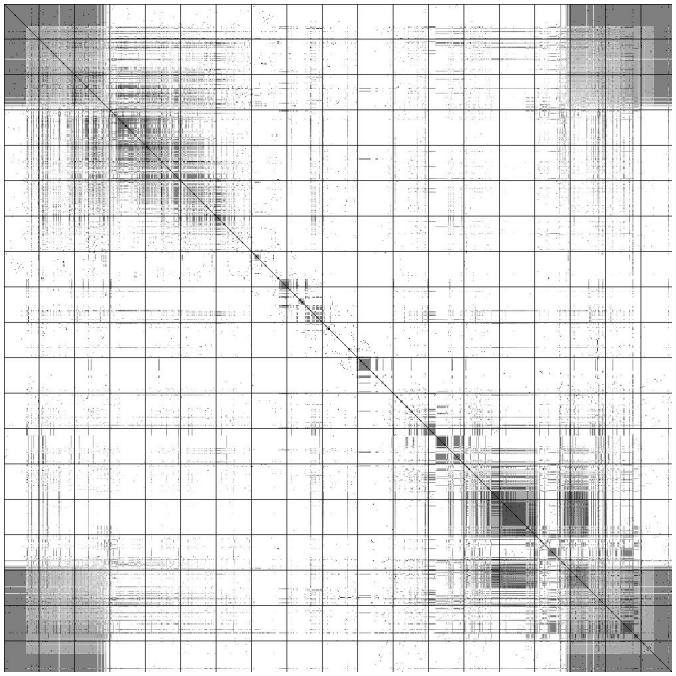


Figura 4 – Representação da matriz do *topological overlap* da rede metabólica da *Synechocystis sp PCC. 6803* após 1000 MCS.

O que fazemos é tratar da força resultante sobre um bloco de nós e não sobre cada nó individualmente. Fazemos isso porque ao movimentar nós individualmente fica impossível retirar o sistema do mínimo local de energia.

Não podemos, entretanto, definir os blocos de nós aleatóriamente, pois correríamos o risco de agrupar nós com pouca relação e de separar grupos de nós muito relacionados - desfazendo, assim, o ordenamento obtido. Para definir os blocos, analisamos cada linha da matriz do *topological overlap* individualmente, como exemplificado nas figuras 5 e 6.

O gráfico de uma linha i da matriz M é basicamente a representação do *topological overlap* entre o nó i com todos os outros nós da rede. Comparando as figuras 5 e 6 vemos que onde aparecem blocos de nós em 5 existem picos no *topological overlap* de i em 6. Para definir os blocos executamos o seguinte algoritmo:

- Escolhe uma linha i de M e percorre todos os elementos j dessa linha.
- Se $To_{i,j} > 0$ adiciona esse nó a um bloco $B_k(n,l)$. Equanto $To_{i,j} \neq 0$, continua adicionando os nós ao mesmo bloco.
- Se $To_{i,j} = 0$, desconsidera o nó j e para de adicionar nós ao bloco $B_k(n,l)$. A partir de j o primeiro nó com $To_{ij} > 0$ define o início de um novo bloco $B_{k+1}(n,l)$.
- Repete esse processo para todas as linhas e retorna todos o blocos obtidos.

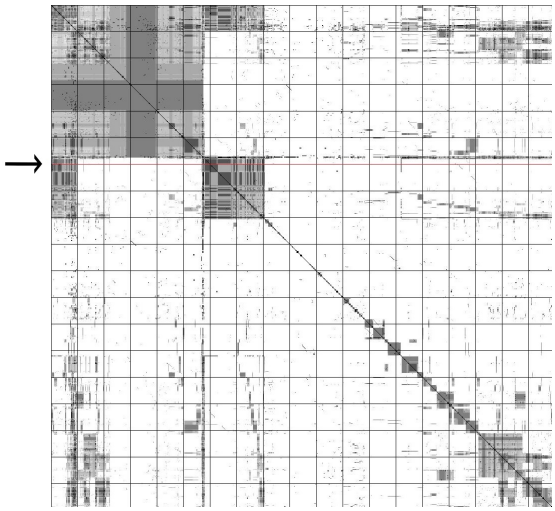


Figura 5 – Para definir os conjuntos de nós analisamos as linhas da matriz do *topological overlap* individualmente. Aqui temos a matriz do *topological overlap* obtida pelo clustering por similaridade para a rede da *Synechocystis sp PCC. 6803* , a seta aponta para a linha representada na figura 6, que também está evidenciada em vermelho (To_{350}).

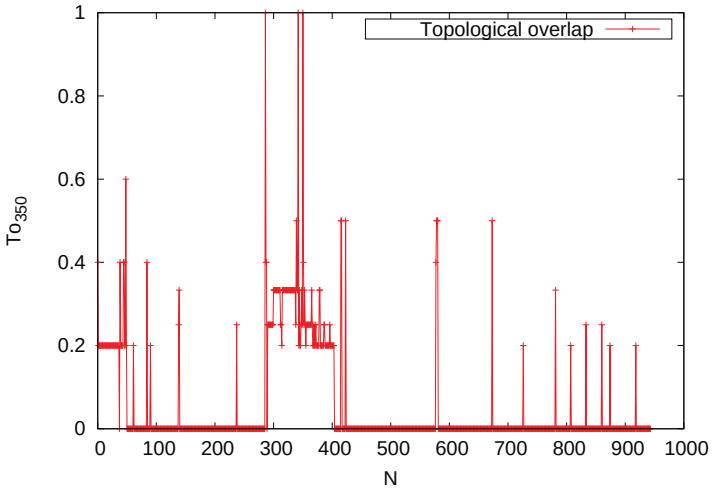


Figura 6 – Representação da linha 350 do *topological overlap* da rede metabólica da *Synechocystis sp PCC. 6803* após a aplicação do nosso método de *clustering por similaridade*. Comparando essa imagem com a figura 5 vemos que os blocos de nós coincidem com picos no gráfico do To_{350}. Os picos entre 0 e 70 representam a interação entre os blocos e o pico entre 300 e 400 representa o próprio bloco onde o nó 350 está contido (figura 7)

.

Usamos $To_{i,j} = 0$ como limite para os blocos pois não gostaríamos de mover nós não correlacionados no ordenamento. Com os blocos definidos, executamos o seguinte algoritmo:

- Encontra todos os blocos presentes na matriz do *topological overlap* da rede.
- Então seleciona um bloco de nós $B_k(n, l)$ e calcula a força resultante sobre esse conjunto F_{Bj}:

$$F_B = \sum F_{R_n} \text{ , para } n \in B_k \tag{2.10}$$

  Se $F_{conj} > 0$ dizemos que seu sentido é para direita, e se $F_{conj} < 0$ seu sentido é para a esquerda.
- Move o conjunto selecionado pelo ordenamento até que a força resultante sobre ele mude de sinal (de sentido).
- Retorna o novo ordenamento.

Na figura 7 temos um exemplo do funcionamento desse algoritmo, que parte do ordenamento da figura 5.

2.4.4 Ordenamento Inicial

Como estamos fazendo um algoritmo heurístico, as soluções deste dependem das condições iniciais escolhidas, por isso, temos que ter cuidado ao defini-lás para obtermos um resultado final que não seja enviesado. A primeira opção é um ordenamento aleatório, que a não ser que sejamos muito sortudos (ou azarados), pouco influenciará no resultado.

Podemos, no entanto, melhorar esse ordenamento retroativamente. Para isso começamos de um ordenamento inicial aleatório e então executamos vários *Monte Carlo Steps* na rede. Ao final dos MCS calculamos separação entre os nós no ordenamento e armazenamos numa matriz X , repetimos esse processo várias vezes (somando as matrizes ao fim de cada repetição) e calculamos a separação média dos nós no ordenamento. Aplicamos o Kruskal nessa matriz de distâncias para que nós com menor separação média fiquem mais próximos no ordenamento inicial.

2.5 Clustering por similaridade

Nesse método também definimos uma medida da semelhança entre dois nós (σ_{ik}) e fazemos isso comparando linhas inteiras da matriz do *topological overlap* (ao invés de

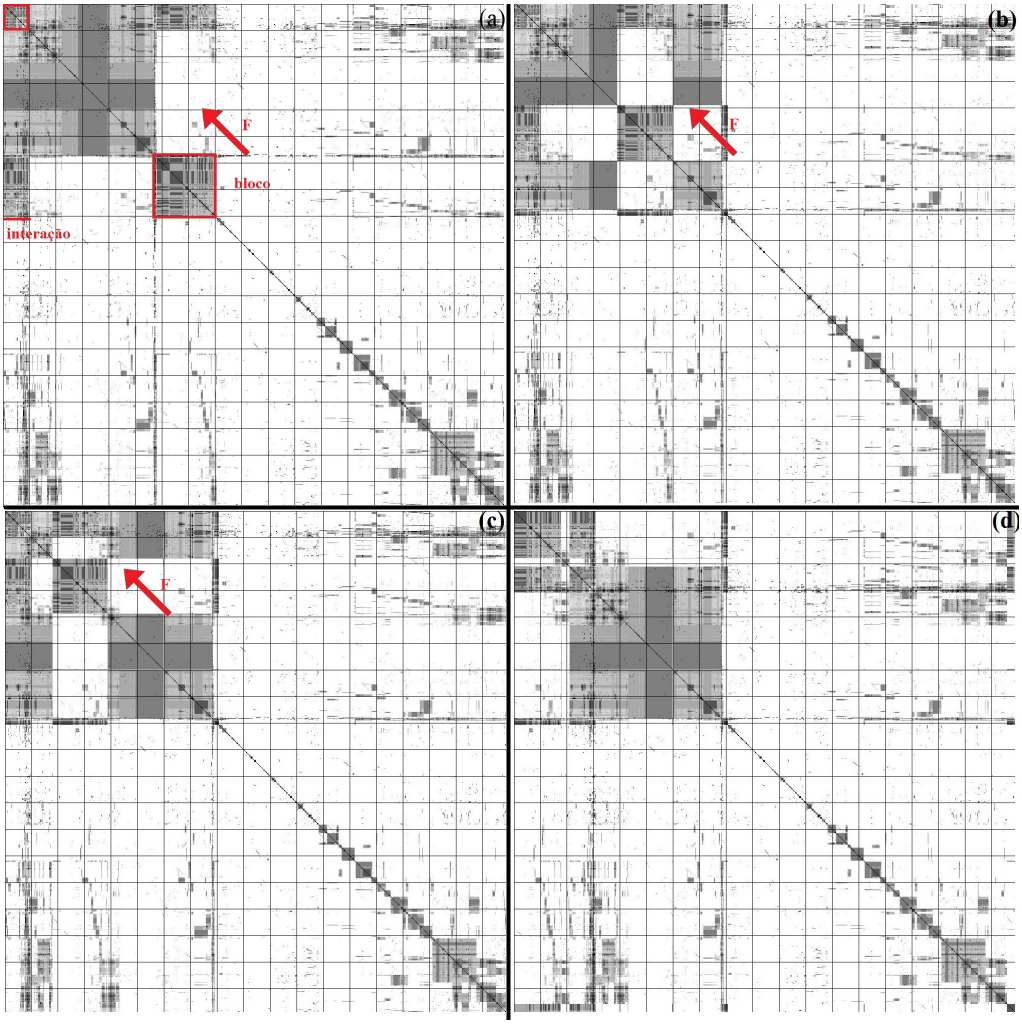


Figura 7 – (a): Existe uma força de atração entre os blocos identificados em vermelho, pois vemos que o *topological overlap* entre esses dois blocos é diferente de zero. Usando as figuras 5 e 6 como referência, identificamos o maior bloco. Em (b) e (c), vemos o bloco sendo arrastado, em (d) temos a inversão no sinal da força sobre o bloco, logo ele para de ser arrastado.

compararmos o To_{ij} entre dois nós, como no Kruskal). Essa função $\sigma_{i,k}$ é definida da seguinte forma:

$$\sigma_{i,k} = \sum_{j=1}^{N} \frac{(To_{ij} - To_{kj})^2}{(To_{ij} + To_{kj} + \varepsilon)^2} \tag{2.11}$$

Onde ε é uma constante positiva ($\varepsilon = 0.000001$) usada para evitar que o denominador seja igual a zero. Quanto menor o valor de σ_{ik} mais próximas as linhas da matriz, isto é, mais parecidos os vizinhos dos nós i e k. O algoritmo utilizado nesse caso se resume em:

- Calcular σ_{ik} entre todas as linhas da matriz do *topological overlap* .
- Aplicar o algoritmo de Kruskal usando σ_{ik} como peso para as ligações.

É importante salientar que nessa formulação queremos a árvore geradora mínima que minimiza o peso de σ_{ik} como proposto originalmente pelo algoritmo de Kruskal.

3 Resultados

Para começar, aplicaremos o algoritmos ao grafo das figuras 1 e 2 (partindo de um ordenamento inicial aleatório). As matrizes resultantes estão representadas na figura 8. Vemos que os algoritmos, sem excessão, agruparam a comunidade formada pelos nós A, B, C, D e L no ordenamento. A maneira que os nós ficam dispostos, no entanto, variou de algoritmo para algoritmo. Esse é um dos desafios que temos ao lidar com esse problema, como evidenciado pelos nós C e D. Eles tem exatamente os mesmos vizinhos na rede, logo como podemos diferenciá-los no ordenamento? Qual é a configuração ideal?

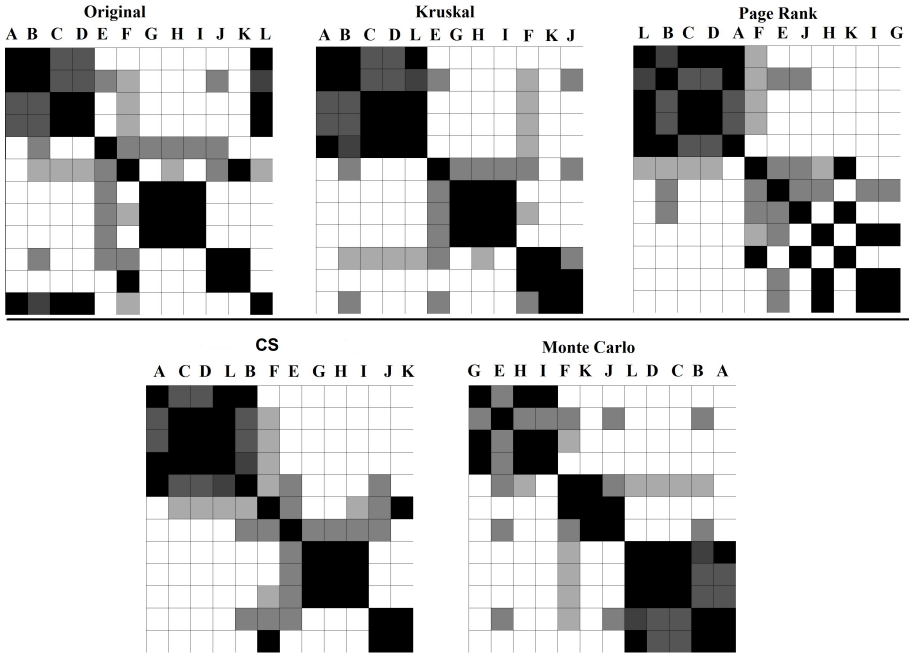


Figura 8 – Comparação do resultado dos algoritmos aplicados sobre o grafo das figuras 1 e 2. Todos os resultados matem a comunidade formada pelos nós A, B, C, D e L unida no ordenamento. A comunidade com os nós G, H, I, no entanto, é dividida pelo Monte Carlo e o PageRank.

Não existe uma maneira de responder essa questão a priori, o que fazemos nesse trabalho é buscar uma maneira de validar os resultados obtidos pelos algoritmos. Isso é possível através da análise das redes PPI, para isso, usamos os termos GO (de *Gene Ontology*) que associam processos biológicos às proteínas. Nesse contexto, cada proteína tem um conjunto de termos GO que simbolizam os processos nos quais ela atua. Por exemplo:

- GO:0003735: *Structural constituent of ribosome.* Isto é, a ação de uma molécula que

contribui para a integridade estrutural do ribossomo.

- GO:0006412: *Traslantion*. Faz parte do processo de tradução, onde uma proteína é formada a partir de uma sequência de mRNA (RNA mensageiro).
- GO:0000287: *Magnesium ion binding*. As proteínas associadas a esse termo GO interagem (ligam-se) com íons magnésio.
- GO:0005886: *Cell membrane.*. Esse termo GO é presente em proteínas que fazem parte da membrana celular.

Para que possamos comparar os ordenamentos obtidos com os termos GO, assumimos uma hipótese fundamental: proteínas que compartilhem um termo GO e, portanto, que fazem parte de um mesmo processo biológico, tem maior probabilidade de interagir entre si. Logo, no ordenamento final esperamos que essas proteínas fiquem próximas.

Com o intuito de quantificar essa informação traçamos um perfil para cada termo GO sobre o ordenamento. O primeiro fator que temos que considerar é o número de vezes que cada termo GO aparece na rede. Não faz sentido analisar o perfil de um termo GO que se relacione apenas com uma proteína da rede, porque essa poderia ocupar qualquer posição no ordenamento. Logo, restringimos nossa análisa aos termos que apareçam em no mínimo 10 proteínas. Para estes, definimos seu perfil ϕ_i como:

$$\phi_i = \sum_{j=1}^{N} \frac{\Delta_j^{go}}{s_{ij}} \tag{3.1}$$

Onde s_ij é a separação dos nós no ordenamento e Δ_j^{go} é pontuação do termo GO na proteína na posição. Caso a proteína i tenha apenas um termo go associado temos que $\Delta_j^{go} = 1$, já para $n > 1$ termos GO $\Delta_j^{go} = \frac{1}{n}$. Essa normalização é feita, porque se uma proteína participar de um único processo consideramos que esta é mais relevante para o mesmo.

Não podemos, obviamente, associar os termos GO as redes metabólicas. Por isso devemos buscar uma forma de avaliar gráficos como os da figura 8. Nessa representação cada célula i e j mostra o *topological overlap* entre os nós, quanto mais escura a célula o To_{ij} é mais próximo de 1 e quanto mais clara mais próximo de 0, portanto, a diagonal principal sempre será escura pois $To_{i=j} = 1$. Qualitativamente, células "pretas"longe da diagonal principal (onde j é muito diferente de i) indicam que os nós i e j interagem entre si, mas que estão separados no ordenamento. Os algoritmos de ordenamento deveriam ser capazes de aproximar esses nós de modo que os nós relacionados se concentrem em torno da diagonal principal.

Como buscamos comparar diversos algoritmos precisamos de uma maneira para descrever esse critério matematicamente. Por sorte, a função $\rho(n)$ que faz justamente isso,

já foi definida por [3]. Ela é descrita como:

$$\rho(n) = \frac{1}{N-n} \sum_{i=1}^{N-n} To_{i,i+n} \tag{3.2}$$

Essa função avalia a probabilidade de dois nós separados por n posições no ordenamento interagirem entre si. Para isso, avalia o número de células escuras presentes em diagonais que distam n da diagonal principal. Nesse traballho analisamos duas redes metabólicas (do *Mycoplasma genitalium G37* e *Synechocystis sp PCC. 6803*) e duas redes PPI (*Candidatus Hodgkinia cicadicola Dseme Mesoplasma florum L1*).

3.1 Redes PPI

3.1.1 *Candidatus Hodgkinia cicadicola Dsem*

A maior componente conexa da rede PPI do *Candidatus Hodgkinia cicadicola Dsem*tem somente 107 nós, com o resultado dos algoritmos confeccionamos os gráficos da figura 9. Como dito anteriormente gostaríamos de obter um perfil que representasse essas matrizes. Numa primeira aproximação definimos uma função W_i, que é dada pela expressão:

$$W_i = \sum_{j=1}^{N} \frac{To_{ij} - \tilde{To}_i}{\sigma_i} e^{-s_{ij}} \tag{3.3}$$

Onde $\tilde{To}_i$ é o *topological overlap* médio da linha i e σ_i é o desvio padrão do To nessa mesma linha. Como sempre s_{ij} é a separação entre os nós i e j no ordenamento. O perfil de cada ordenamento esta desenhado sob a sua respectiva matriz M (figura 9 de (a) a (g)). Um pico no gráfico de W_i mostra uma maior densidade no *topological overlap* nas vizinhanças dessa linha em M, equanto um vale identifica uma região menos densa. O perfil do ordenamento aleatório (mostrado em (a)) tem picos menores (é necessário salientar que os gráficos não estão na mesma escala) e em menor quantidade que o obtido pelos algoritmos, o que nos diz que os algoritmos efetivamente estão aproximando os nós no ordenamento, pois geram gráficos mais densos.

Na figura 10 temos o gráfico em escala semilog da função $\rho(n)$ com a distância n de diagonal principal. O primeiro detalhe que observamos é que os processos de Monte Carlo tendem a concentrar os nós em torno da diagonal principal, por isso temos um $\rho(n)$ maior que os demais para $n < 30$. A partir desse ponto, $\rho(n)$ decaí exponencialmente, o que evidencia que existem poucos nós que interagem entre si e que estão separados no ordenamento (para nós longe da diagonal principal a diferença ente i e j é grande). Também vemos que as variações na figura 9 (e), (f) e (g) pouco interferem no gráfico de $\rho(n)$. Por último observamos que o método de *clustering por similaridade* (CS) desenvolvido por nós tem pequenas diferenças do Kruskal (KR).

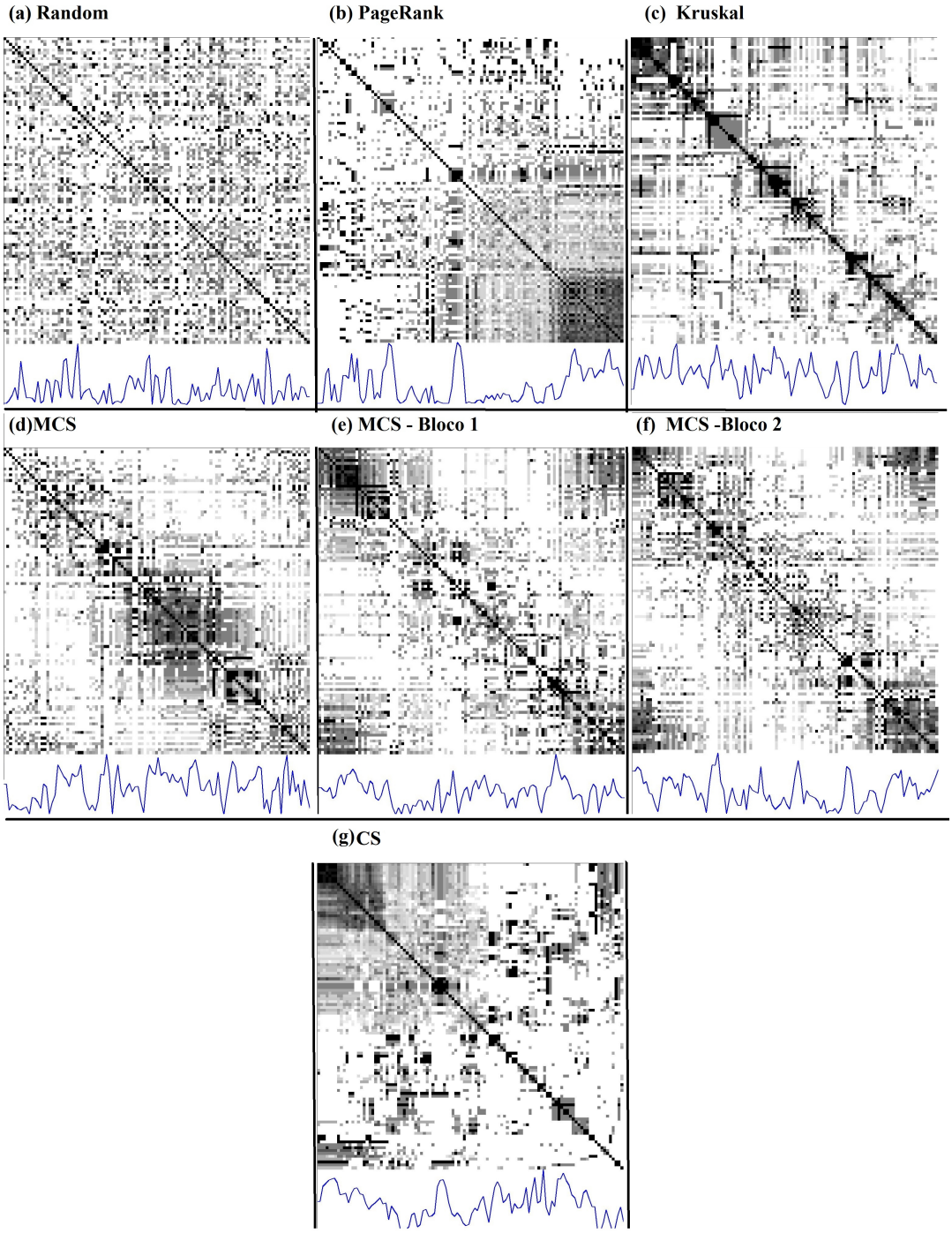


Figura 9 – Resultados dos algoritmos para a rede do *Candidatus Hodgkinia cicadicola Dsem*. Em (d) foram feitos 1000 MCS, em (e) partimos de 1000 MCS,executamos 30 permutações com blocos de nós e por fim mais 400 MCS. Já em (f) intercalamos cada passo com Blocos com 200 MCS, também partindo de 1000 MC. Vemos que todos os métodos apresentam mais picos e picos maiores que o ordenamento aleatório o que indica que os algotitmos de (b) a (g) estão efetivamente aproximando os nós.

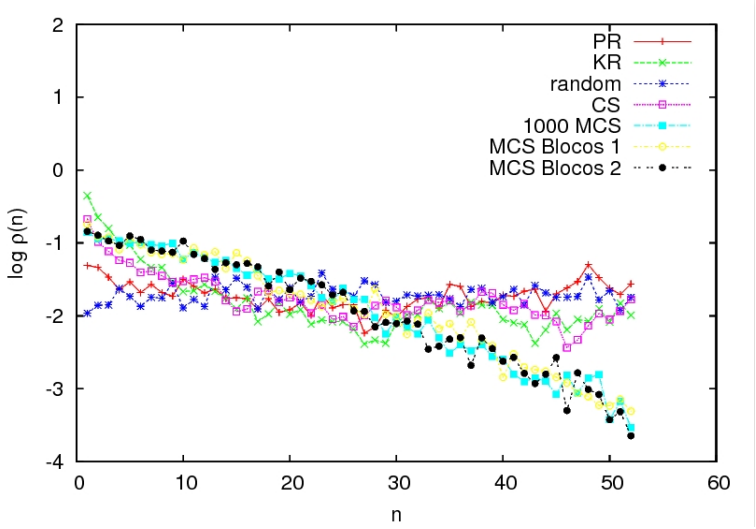


Figura 10 – Grafíco de $\rho(n)$ em escala semilog para todos os algoritmos. Vemos que os processos de Monte Carlo tendem a agrupar os nós em torno da diagonal principal, o que indica que existem poucos nós que interagem entre si e que estejam muito separados no ordenamento. O *PageRank* (PR) dá um resultado levemente superior ao aleatório e ambos o PR, o Kruskal (KR) e o método de *clustering por similaridade* (CS) mostram uma distribuição mais difusa pois $\rho(n)$ aumenta para $n > 30$.

Comparar nossos resultados com as funções celulares do organismo é comparar os perfis gerados por W_i com os perfis dos termos GO, portanto fica clara a importância da definição de W_i. Por isso, para efeitos de comparação, definimos duas funções W_i' e W_i'' que também criam perfis a partir da matriz do *topological overlap* .

$$W_i' = \sum_{j=1}^{N} \frac{To_{ij} - \tilde{To}_i}{\sigma_i s_{ij}} \tag{3.4}$$

Os picos e vales devem ser mais atenuados por essa definição, pois W_i' não decresce exponencialmente com a separação entre os nós. Para W_i'' aplicamos o mesmo algoritmo que define os blocos sessão (2.4.3), nesse caso a altura do perfil é dada pelo *topological overlap* dos nós.

Ao fazermos os gráficos dos perfis W_i, W_i' e W_i'' contra os termos GO (somente 6 termos go estão presentes em mais de 10 proteínas nesse organimo, que são:0003735, 0005840, 0006412, 0016021, 0005737 e 0005524) para cada algoritmo não observamos uma forte correlação R entre as curvas. Não há nem mesmo um padrão para o desempenho dos algoritmos. Por exemplo, o algoritmo de Kruskal que encontra a maior correlação para os termos 0005840, 0003735 e 0006412 ($R = 0.37$, $0,26$ e $0,25$), obtêm uma correlação negativa para os três termos GO restantes. Combinando todos os algoritmos e os três perfis, os melhores resultados obtidos estão na tabela 3.1.1 .

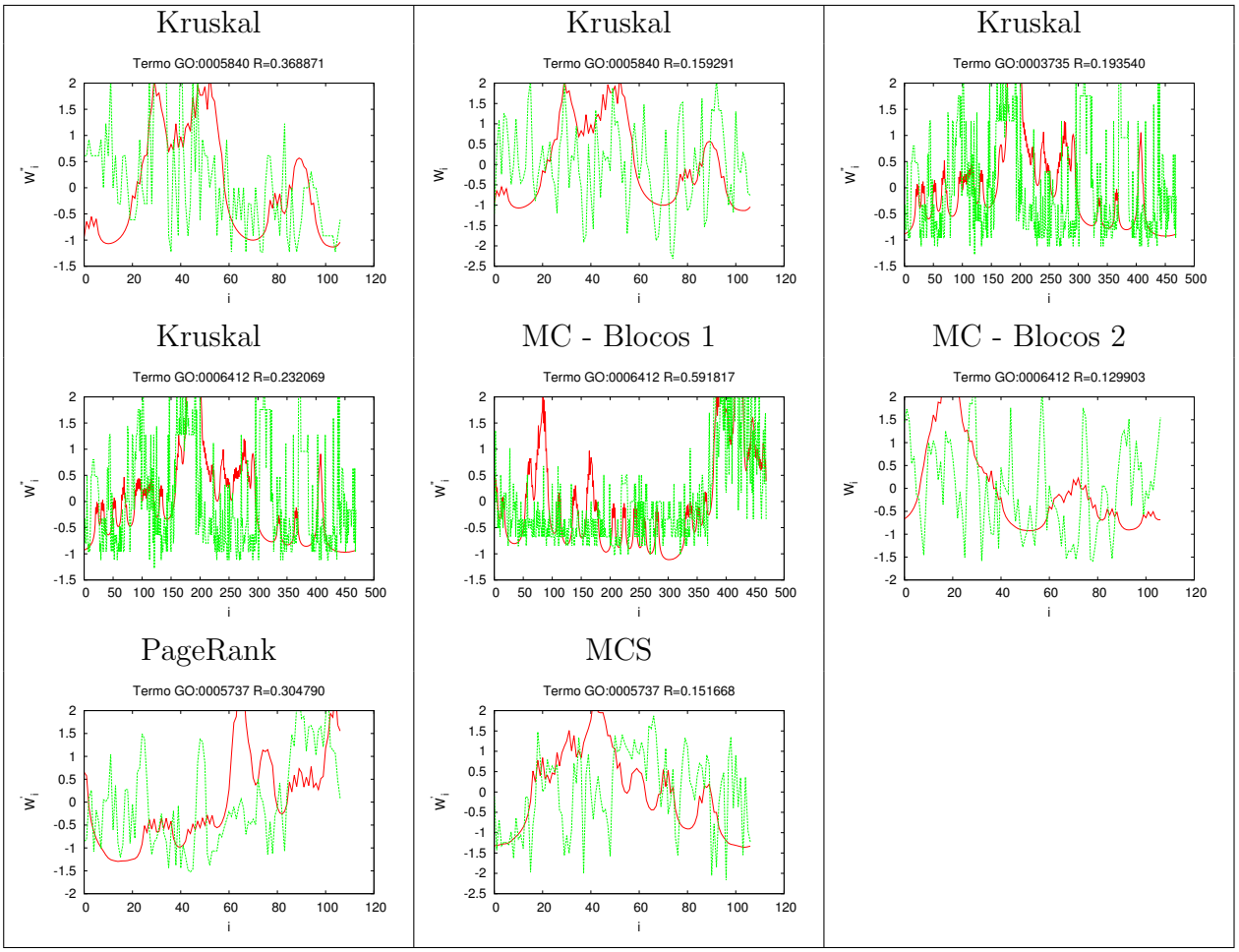


Figura 11 – Perfis com maior correlação para a rede PPI do *Candidatus Hodgkinia cicadicola Dsem*. Nenhum algoritmo obteve um desempenho constante para todos os termos GO. Os melhores resultados foram obtidos pelo Kruskal com o perfil W_i''‘.

Dentre os algoritmos o Kruskal obteve os melhores resultados, atigindo uma correlação de $R = 0.37$, equanto o nosso método de *clustering por similaridade* não obteve nenhum resultado expressivo. Ao analisarmos as imagens da matriz do *topological overlap* (figura 9 e o gráfico de $\rho(n)$ (figura 10) vemos que os métodos de Monte Carlo estão efetivamente aproximando os nós com maior *topological overlap* em comum. Isso, somado ao fato de que as correlações obtidas foram pequenas para esses métodos, indica que a maneira que definimos os perfis W_i, W_i' e W_i'' talvez não seja a ideal.

3.1.2 *Mesoplasma florum L1*

A rede PPI do *Mesoplasma florum L1*contem 470 nós na sua maior componente conexa, onde 27 termos GO aparecem em pelo menos 10 proteínas. Como no caso do *Candidatus Hodgkinia cicadicola Dsem*os algoritmos produziram resultados que variaram

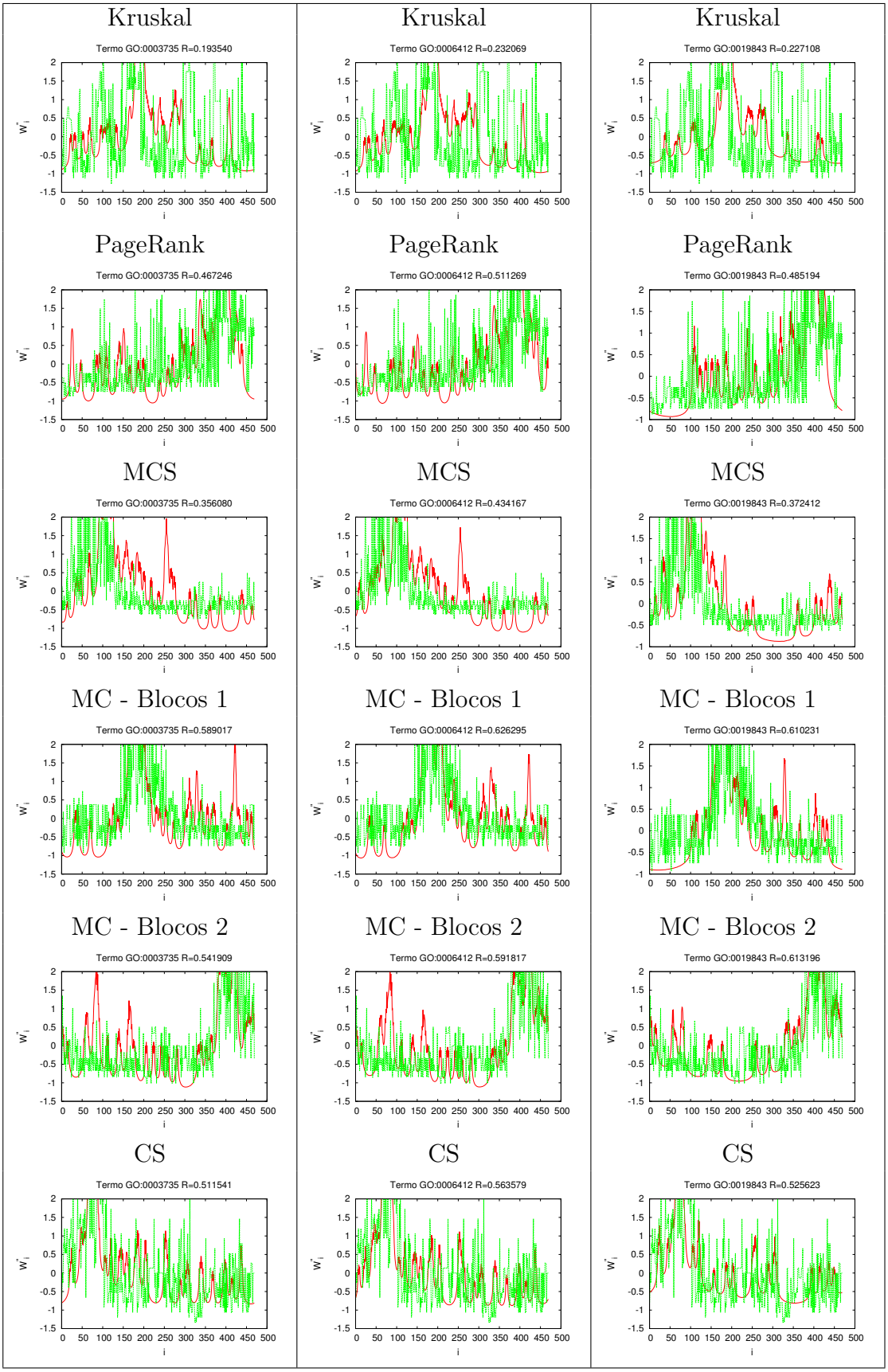
fortemente entre os 27 termos GO análisados. Destes, três termos de códigos, 0003735, 00006412 e 0019843, respectivamente, tiveram as maiores correlações R como representado na figura 12. O perfil W_i'' (em verde), apresentou os melhores resultados em todos os casos.

O algoritmo de Monte Carlo sozinho apresentou resultados inferiores as variações com blocos de nós. Devido a grande variação na correlação obitda para os diferentes termos GO, não podemos afirmar que isso é de fato um aspecto geral. O método de clustering por similaridade (CS) conseguiu resultados similares ao dos MCS - Blocos 1 e 2, ao contrário do que ocorreu para a rede do *Candidatus Hodgkinia cicadicola Dsem*.

Nenhum dos algoritmos encontrou todos os termos GO consistentemente, talvez a maneira como os perfis tenham sido definidos não seja a ideal, ou a maneira que definimos os blocos de nós também possa ser aprimorada. Ainda assim encontramos correlações maiores que 60% para alguns termos,uma nova maneira de analisarmos os perfis como feita em [3] poderia melhorar os resultados obtidos.

No que tange as matrizes do *topological overlap* obtidas, os resultados estão nas figuras 13 e 14. Como nas outras redes, os algoritmos de Monte Carlo aproximam os nós da diagonal principal, logo, seriam melhores soluções para o MBMP, devido a isso a função $\rho(n)$ é maior para $18 < n < 100$ e menor para $n > 150$. O desempenho do clustering por similaridade é bastante similar ao do Kruskal, sendo menos denso para $n < 25$ e mais denso $n > 200$. Novamente o PageRank é levemente superior ao ordenamento aleatório, o que é o esperado visto que a função primordial desse algoritmo é ordenar os nós a partir uma relevãncia e não evidenciar as comunidades.

Figura 12 – Perfis com maior correlação para a rede PPI do *Mesoplasma florum L1*, nessa rede as maiores correlações ocorreram para três termos GO específicos:0003735, 00006412 e 0019843.



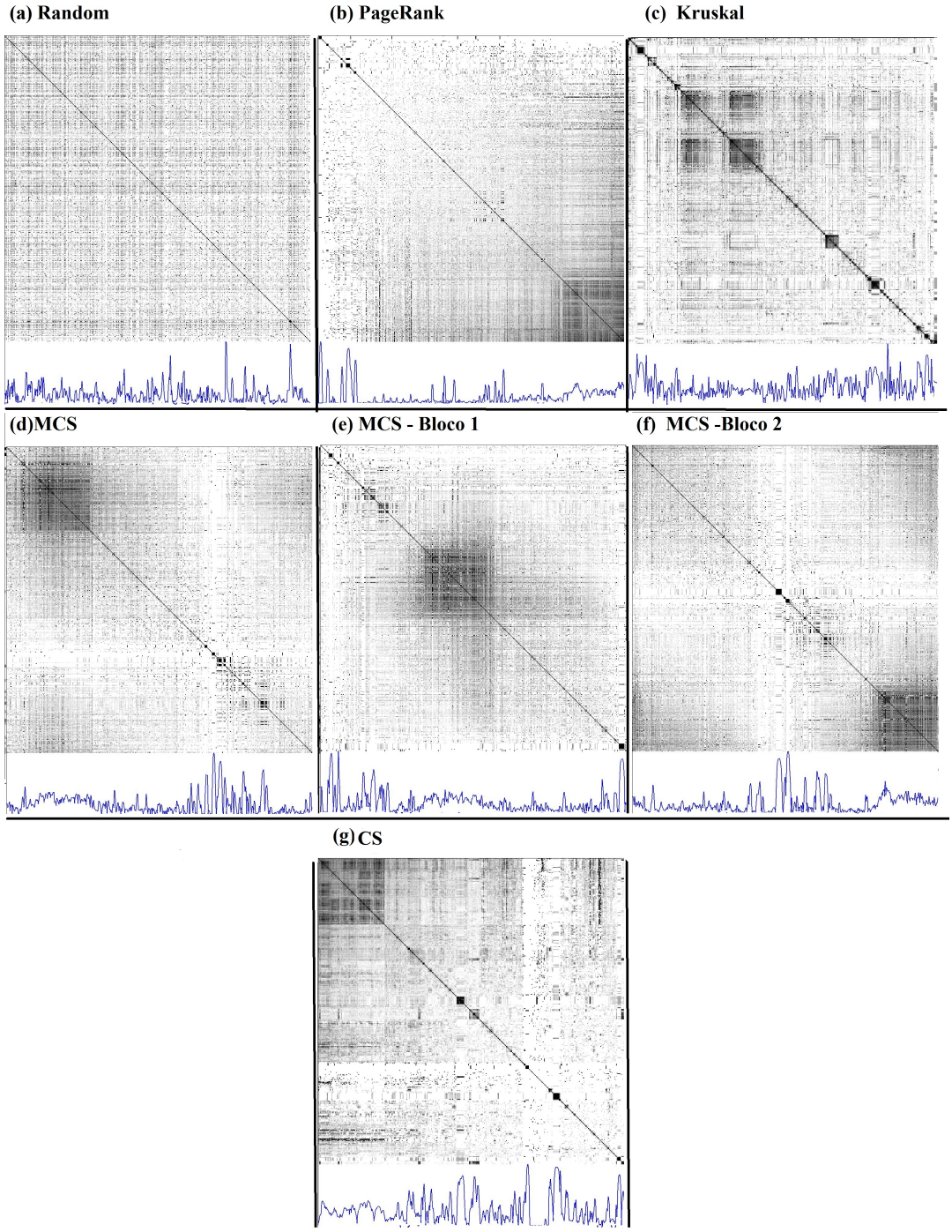


Figura 13 – Resultados dos algoritmos para a rede do *Mesoplasma florum L1*. Resultados dos algoritmos para a rede metabólica do *Mycoplasma genitalium G37* . Em todas as imagens partimos de um ordenamento inicial aleatório e o perfil calculado por W_i. Em (a) temos o ordenamento inicial. Em (b) o PageRank. Em (c) o Kruskal. Em (d) foram feitos 1000 MCS, em (e) 1000 MCS então 30 permutações com blocos e mais 400 MCS, por fim em (f) 1000 MCS e então intercalamos cada passo com Blocos com 200 MCS.

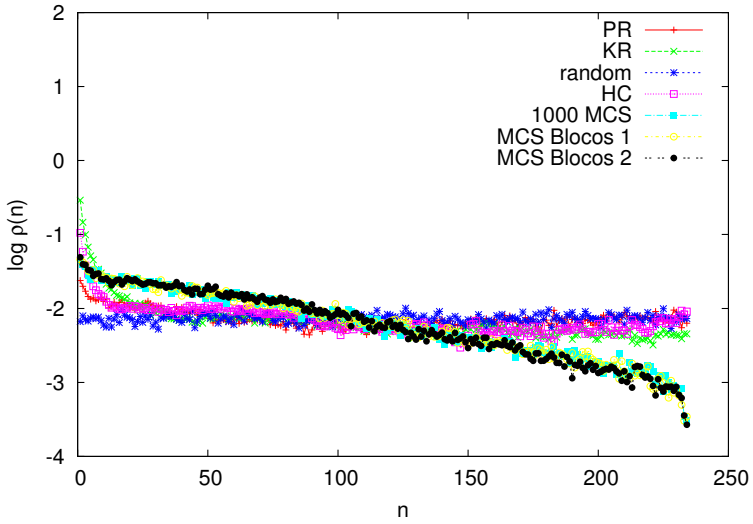


Figura 14 – Grafíco de $\rho(n)$ em escala semilog para todos os algoritmos. Vemos que os processos de Monte Carlo tendem a agrupar os nós em torno da diagonal principal, por isso a maior probabilidade de existirem nós interagentes para $n < 225$. O *PageRank* (PR) dá um resultado levemente superior ao aleatório, mas é superado pelo Kruskal (KR) para ($n > 150$). Ao contrário do que ocorre para o *Candidatus Hodgkinia cicadicola Dsemo* CS mostra uma melhora significativa no desempenho do Kruskal para $n > 200$.

3.2 Redes Metabólicas

3.2.1 *Synechocystis sp PCC. 6803*

Na rede metabólica da *Synechocystis sp PCC. 6803* temos 944 nós. Para testar os algoritmos nessa rede usamos dois ordenamentos iniciais: o aleatório e o descrito em 2.4.3.1 Para o último fazemos 200 repetições com 200 MCS cada (ordenamento inicial representado na figura 16). A figura 15 representa o resultado os algoritmos aplicados sobre esses ordenamentos, em ambos os casos graficamos também o perfil W_i descrito em 3.1.1.

Comparando as figuras 16 e 15 vemos que os ordenamentos finais em 15-(d),(e) e (f) são muito parecidos com o ordenamento inicial, isso evidencia que as distâncias médias obtidas através das repetidas iterações são boas aproximações para as posições de equilíbrio dos nós no ordenamento. Também, ao compararmos os resultados dos MCS para os dois ordenamentos iniciais, fica clara a presença de estruturas análogas o que, por sua vez, indica que o sistema está convergindo para um estado final similar.

As figuras 17 e 18 trazem os gráficos de $\rho(n)$. Novamente vemos que os ordenamentos obtidos pelo processo de Monte Carlo aglomeram os nós no entorno da diagonal principal, de modo que para $n < 225$ temos uma probabilidade $\rho(n)$ grande de encontrarmos nós que interagem entre si, mas a medida que n aumenta essa probabilidade cai exponencialmente.

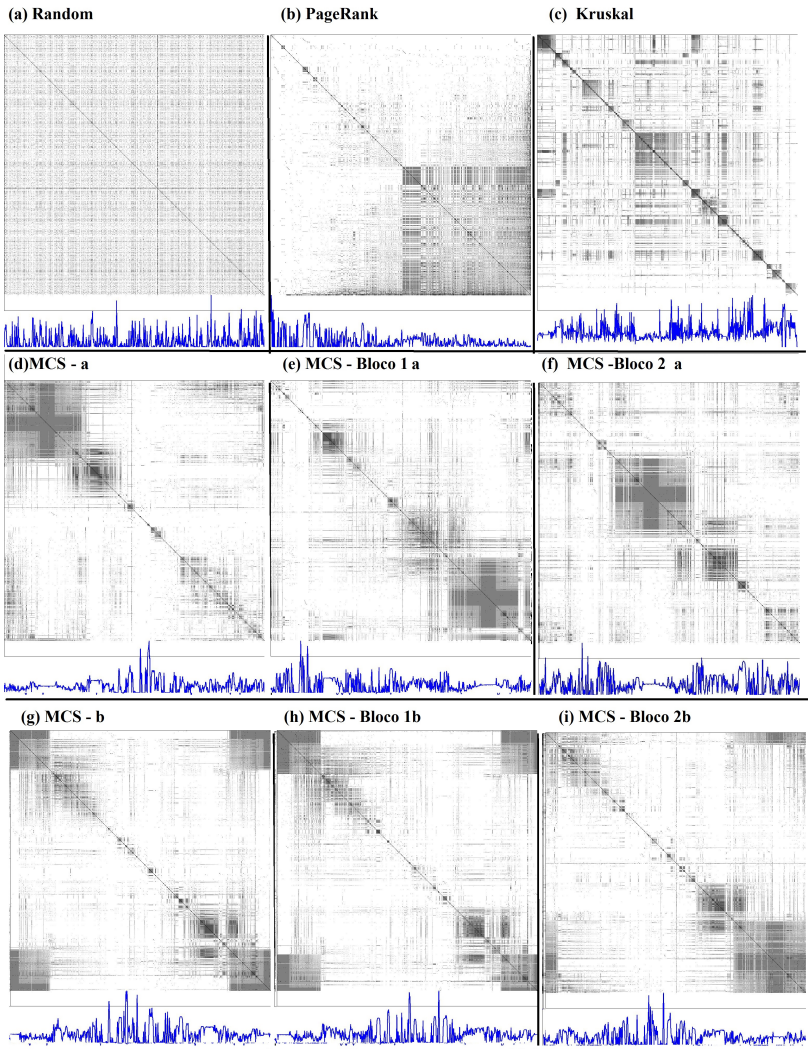


Figura 15 – Resultados dos algoritmos para a rede da *Synechocystis sp PCC. 6803* . As figuras (d),(e) e (f) partiram do ordenamento inicial não-aleatório. Para isso fazemos um Kruskal sobre as distâncias médias dos nós após 200 repetições de 200 MCS cada. Em (d) foram feitos 1000 MCS, em (e) partimos de 1000 MCS,executamos 30 permutações com blocos de nós e por fim mais 400 MCS. Já em (f) intercalamos cada passo de Blocos com 200 MCS, também partindo de 1000 MC. Em (g), (h) e (i) fazemos o mesmo processo que em (d), (e) e (f), respectivamente, mas partimos de um ordenamento inicial aleatório.

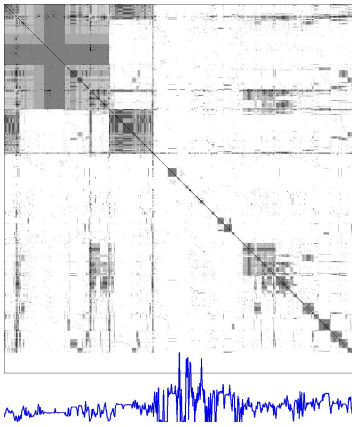


Figura 16 – Ordenamento inicial, obtido através do Kruskal sobre a matriz das distâncias médias dos nós após 200 repetições do algoritmo com 200 MCS cada.

Para a *Synechocystis sp PCC. 6803* , o algoritmo de *clustering por similaridade* apresenta um resultado superior ao do Kruskal para $n > 200$.

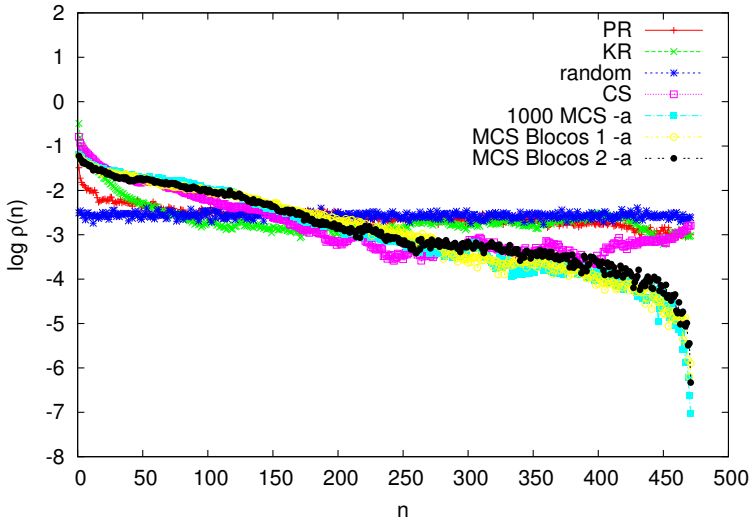


Figura 17 – Grafíco de $\rho(n)$ em escala semilog para todos os algoritmos. Vemos que os processos de Monte Carlo tendem a agrupar os nós em torno da diagonal principal, por isso a maior probabilidade de existirem nós interagentes para $n < 225$. O *PageRank* (PR) dá um resultado levemente superior ao aleatório, mas é superado pelo Kruskal (KR) para ($n > 150$). Ao contrário do que ocorre para o *Candidatus Hodgkinia cicadicola Dsemo* CS mostra uma melhora significativa no desempenho do Kruskal para $n > 200$.

Na figura 18 temos a comparação entre os processos de Monte Carlo com diferentes ordenamentos iniciais. Vemos que não há uma grande variação em $\rho(n)$, no entanto, as condições iniciais não aleatórias parecem resultar num gráfico mais difuso, por isso, temos um leve aumento na função $\rho(n)$.

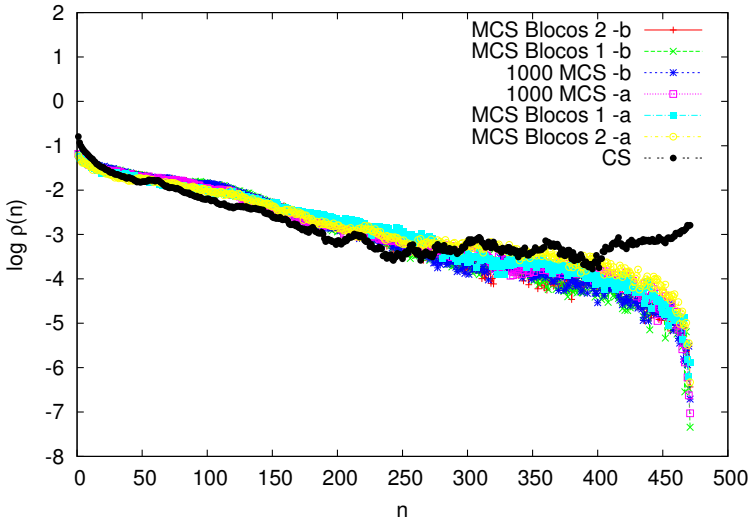


Figura 18 – Grafíco de $\rho(n)$ em escala semilog para o processos de Monte Carlo com diferentes condições iniciais. O grupo (a) tem ordenamento inicial não-aleatório, enquanto o grupo (b) parte de condições iniciais aleatórias. Observamos que o grupo não aleatório forma um gráfico levemente mais difuso devido ao aumento em $\rho(n)$

3.2.2 *Mycoplasma genitalium G37*

A rede metabólica do *Mycoplasma genitalium G37* tem 381 nós. Para esse organismo partimos de um ordenamento inicial aleatório. A figuras 19 e 20 trazem dos gráficos das matrizes do *topological overlap* do *Mycoplasma genitalium G37* e o gráfico de $\rho(n)$ comparando os ordenamentos, respectivamente.

Os algoritmos, novamente, mostram resultados melhores que um ordenamento aleatório, como evidenciando pelos perfis na figura 19 e pela função $\rho(n)$ na figura 20. Também notamos que os algoritmos de Monte Carlo - com blocos , ou sem - apresentam pouca variação entre si ao serem analisados por $\rho(n)$ e que eles tendem a aproximar os termos não-nulos da matriz do *topological overlap* ao redor da diagonal principal, por isso um aumento em $\rho(n)$ para n pequeno. Isso vai de encontro com o que queremos como solução para o MBMP.

Para essa rede metabólica, o clustering por similaridade apresentou melhores resultados que o Monte Carlo nos seguintes intervalos: $140 < n < 170$ e $95 < n < 115$. Esse comportamento não foi observado em nenhuma das outras redes, mesmo em *Mesoplasma florum L1*que tem um tamanho similar. Portanto, podemos atribuir esse resultado a alguma característica específica da rede do *Mycoplasma genitalium G37* .

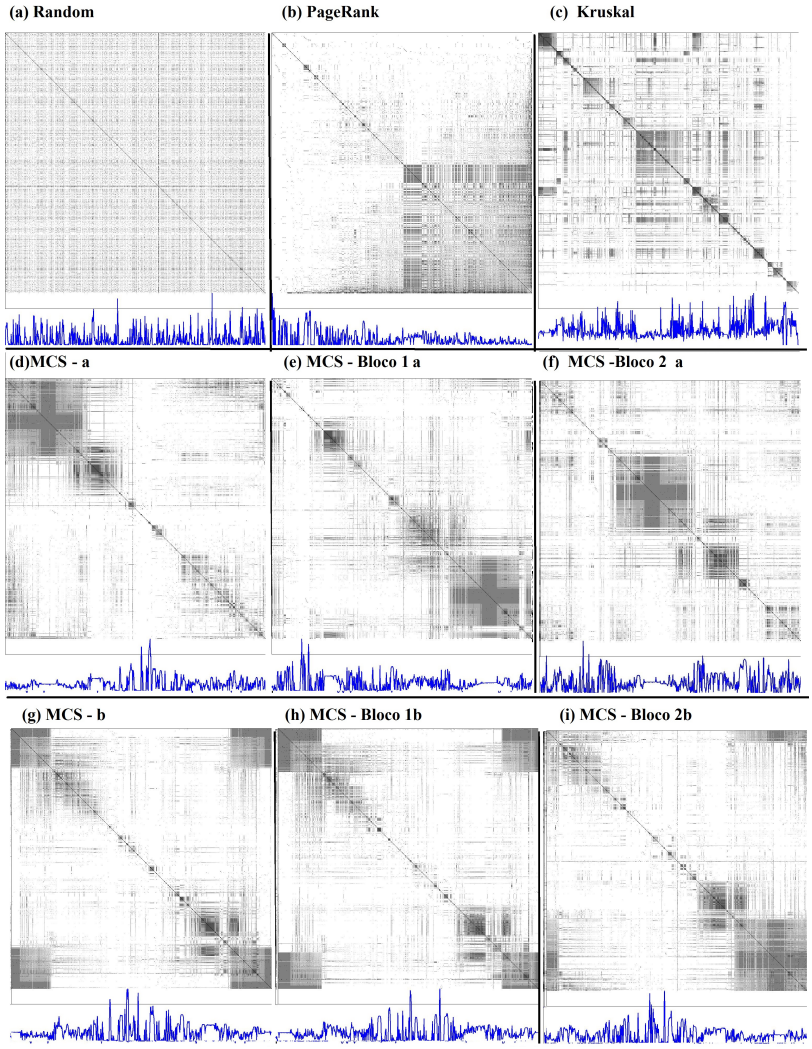


Figura 19 – Resultados dos algoritmos para a rede metabólica do *Mycoplasma genitalium G37* . Em todas as imagens partimos de um ordenamento inicial aleatório e o perfil calculado por W_i. Em (a) temos o ordenamento inicial. Em (b) o PageRank. Em (c) o Kruskal. Em (d) foram feitos 1000 MCS, em (e) 1000 MCS então 30 permutações com blocos e mais 400 MCS, por fim em (f) 1000 MCS e então intercalamos cada passo com Blocos com 200 MCS.

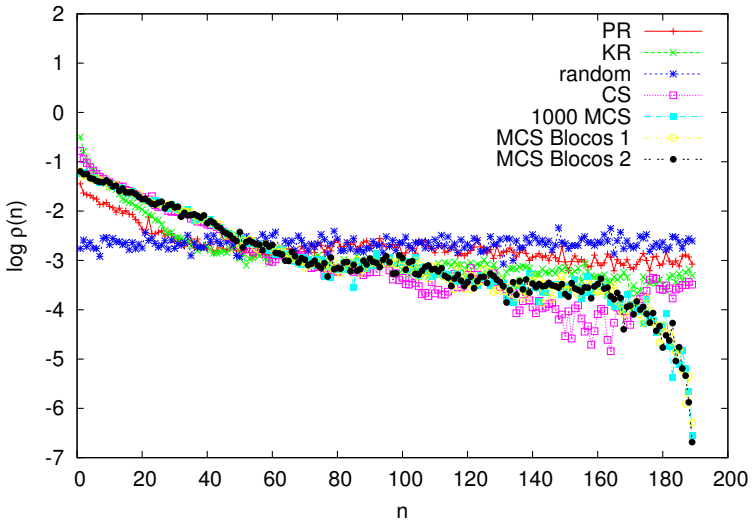


Figura 20 – Gráfico em escala semilog de $\rho(n)$ para os diferentes algoritmos. Vemos que o PageRank só mostra resultados superiores ao Kruskal para nós muito proximos a diagonal principal ($n < 40$), que os algoritmos de Monte Carlo são superiores aos dois a partir de ($n > 100$). Nesse caso, no entanto, o clustering por similaridade (CS) obtêm melhores resultados entre todos os algoritmos para $140 < n < 170$ e $95 < n < 115$.

4 Conclusão

Conforme observado nas figuras 9, 13, 15 e 19 bem como em 10, 14 e 17 e 20 todos os algoritmos analisados obtém resultados melhores que o de um ordenamento aleatório para as redes metabólicas e PPI (dos organismos:*Synechocystis sp PCC. 6803* e *Mycoplasma genitalium G37* , *Candidatus Hodgkinia cicadicola Dseme Mesoplasma florum L1*, respetivamente). Isso é evidenciado pela quantidade e altura dos picos nos perfis de 9, 13, 15 e 19, bem como pela probabilidade de que dois nós que interagem entre si estejam separados por uma distância n no ordenamento dada por $\rho(n)$ [3] em 10, 14, 17 e 20.

Os algoritmos de Monte Carlo tendem a aglomerar os nós no entorno da diagonal principal, por isso, para $n < 225$ e $n < 85$, no caso das redes metabólicas, e $n < 30$ e $n < 125$, no caso das redes PPI, vemos que $\rho(n)$ assume valores maiores que os do PageRank, Kruskal e *Clustering por similaridade*. No entanto, passados esses limites a probabilidade se torna menor que a obtida pelos outros métodos. A introdução dos blocos nos algoritmos de Monte Carlo, bem como a mudança de ordenamento inicial, pouco pareceram influenciar nos ordenamentos finais obtidos (figuras 10, 17 e 18).

O PageRank obtem resultados superiores ao Kruskal somente para nós bem próximos da diagonal principal, isso ocorre pois há uma direcionalidade clara nesse algoritmo. Fica evidente que linhas da matriz M com menor valor de *topological overlap* (isto é, a soma dos *topological overlap* sobre toda a linha da matriz) tendem ter um PageRank menor e a ficar à esquerda no ordenamento, já linhas cuja soma é maior tendem a ter um PageRank maior e a ficarem à direita no ordenamento (por isso vemos um pico em $n > 45$ na figura 10). Além disso a principal função desse algoritmo é ordenar os nós de acordo com a sua relevância e não necessariamente encontrar as comunidades nesse processo, por isso seus resultados foram pouco superiores ao do ordenamento aleatório.

O algortimo de Clustering por Similaridade conseguiu resultados bem próximos ao do Kruskal na maioria dos casos, como o próprio Kruskal faz parte da execução do algoritmo esse resultado não foge do esperado. Na rede do *Mycoplasma genitalium G37* no entanto nos intervalos entre $140 < n < 170$ e $95 < n < 115$ (figura 20) o clustering por similaridade é superior ao prórpio processo de Monte Carlo. Esse comportamento não foi visto em nenuhma das outras redes, logo deve ser causado por alguma característica especiao do *Mycoplasma genitalium G37* .

Ao tentarmos associar os ordenamentos encontrados para as redes *Candidatus Hodgkinia cicadicola Dseme Mesoplasma florum L1*com suas funções celulares, nenhum dos algoritmos obteve resultados consistentes ao comparar os perfis gerados com os termos GO 12 e 3.1.1. No caso da *Mesoplasma florum L1*conseguimos obter correlações $R > 50\%$

para três termos GO distintos 12. Em ambos os casos as maiores correlações ocorreram para os perfis gerados por W_i'' o que indica que essa definição deve estar mais próxima da ideal.

Como a partir da análise de $\rho(n)$ vemos que os algoritmos efetivamente agrupam os nós, parece mais provável que as funções W_i, W_i' e W_i'' (equações 3.3 e 3.4, respectivamente) não sejam ótimas para fazer os perfis das redes. Uma possível solução seria implementar a função usada em [3]. No entanto, a possibilidade de que os agrupamentos obtidos simplesmente não reflitam as funções celulares e que não pode ser descartada a priori. Nesse cenário teríamos encontrado as limitações dos nossos algortimos.

Referências

[1] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. U.S.A.*, 99(12):7821–7826, Jun 2002.

[2] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[3] J. L. Rybarczyk-Filho, M. A. Castro, R. J. Dalmolin, J. C. Moreira, L. G. Brunnet, and R. M. de Almeida. Towards a genome-wide transcriptogram: the Saccharomyces cerevisiae case. *Nucleic Acids Res.*, 39(8):3005–3016, Apr 2011.

[4] Roger Guimera and Luis A Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005.

[5] Fredrik Liljeros, Christofer R Edling, Luis A Nunes Amaral, H Eugene Stanley, and Yvonne Åberg. The web of human sexual contacts. *Nature*, 411(6840):907–908, 2001.

[6] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[7] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[8] Minoru Kanehisa and Susumu Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, 2000.

[9] R. Reyes, D. Gamermann, A. Montagud, D. Fuente, J. Triana, J. F. Urchueguia, and P. F. de Cordoba. Automation on the generation of genome-scale metabolic models. *J. Comput. Biol.*, 19(12):1295–1306, Dec 2012.

[10] A. Franceschini, D. Szklarczyk, S. Frankild, M. Kuhn, M. Simonovic, A. Roth, J. Lin, P. Minguez, P. Bork, C. von Mering, and L. J. Jensen. STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res.*, 41(Database issue):D808–815, Jan 2013.

[11] Rafael Martí, Vicente Campos, and Estefanía Piñana. A branch and bound algorithm for the matrix bandwidth minimization. *European Journal of Operational Research*, 186(2):513–528, 2008.

[12] Ch H Papadimitriou. The np-completeness of the bandwidth minimization problem. *Computing*, 16(3):263–270, 1976.

[13] Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986, 1984.