Metaheurísticas para o problema de agrupamento de dados em grafo

Mariá Cristina Vasconcelos Nascimento

Metaheurísticas para o problema de agrupamento de dados em grafo

Mariá Cristina Vasconcelos Nascimento

Orientador: ***Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho***

Co-orientadora: ***Profa. Dra. Franklina Maria B. de Toledo***

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências - Ciências de Computação e Matemática Computacional.

USP – São Carlos
Fevereiro/2010

Agradecimentos

Agradeço ao Lucas e ao Bob pelo apoio diário, carinho e paciência durante esse período. Agradeço também aos meus pais.

Agradeço a todos professores, amigos e funcionários da USP, que de alguma forma, participaram deste trabalho. Agradeço, em especial, aos meus orientadores André e Franklina.

Agradeço à FAPESP, pelo apoio financeiro.

Agradeço a Deus, pela vida.

Resumo

O problema de agrupamento de dados em grafos consiste em encontrar *clusters* de nós em um dado grafo, ou seja, encontrar subgrafos com alta conectividade. Esse problema pode receber outras nomenclaturas, algumas delas são: problema de particionamento de grafos e problema de detecção de comunidades. Para modelar esse problema, existem diversas formulações matemáticas, cada qual com suas vantagens e desvantagens. A maioria dessas formulações tem como desvantagem a necessidade da definição prévia do número de grupos que se deseja obter. Entretanto, esse tipo de informação não está contida em dados para agrupamento, ou seja, em dados não rotulados. Esse foi um dos motivos da popularização nas últimas décadas da medida conhecida como *modularidade*, que tem sido maximizada para encontrar partições em grafos. Essa formulação, além de não exigir a definição prévia do número de *clusters*, se destaca pela qualidade das partições que ela fornece. Nesta Tese, metaheurísticas *Greedy Randomized Search Procedures* para dois modelos existentes para agrupamento em grafos foram propostas: uma para o problema de maximização da modularidade e a outra para o problema de maximização da similaridade intra-*cluster*. Os resultados obtidos por essas metaheurísticas foram melhores quando comparadas àqueles de outras heurísticas encontradas na literatura. Entretanto, o custo computacional foi alto, principalmente o da metaheurística para o modelo de maximização da modularidade.

Com o passar dos anos, estudos revelaram que a formulação que maximiza a modularidade das partições possui algumas limitações. A fim de promover uma alternativa à altura do modelo de maximização da modularidade, esta Tese propõe novas formulações matemáticas de agrupamento em grafos com e sem pesos que visam encontrar partições cujos *clusters* apresentem alta conectividade. Além disso, as formulações propostas são capazes de prover partições sem a necessidade de definição prévia do número de *clusters*. Testes com centenas de grafos com pesos comprovaram a eficiência dos modelos propostos. Comparando as partições provenientes de todos os modelos estudados nesta Tese, foram observados melhores resultados em uma das novas formulações propostas, que encontrou partições bastante satisfatórias, superiores às outras existentes, até mesmo para a de maximização de modularidade. Os resultados apresentaram alta correlação com a classificação real dos dados simulados e reais, sendo esses últimos, em sua maioria, de origem biológica.

Abstract

Graph clustering aims at identifying highly connected groups or clusters of nodes of a graph. This problem can assume others nomenclatures, such as: graph partitioning problem and community detection problem. There are many mathematical formulations to model this problem, each one with advantages and disadvantages. Most of these formulations have the disadvantage of requiring the definition of the number of clusters in the final partition. Nevertheless, this type of information is not found in graphs for clustering, i.e., whose data are unlabeled. This is one of the reasons for the popularization in the last decades of the measure known as modularity, which is being maximized to find graph partitions. This formulation does not require the definition of the number of clusters of the partitions to be produced, and produces high quality partitions. In this Thesis, *Greedy Randomized Search Procedures* metaheuristics for two existing graph clustering mathematical formulations are proposed: one for the maximization of the partition modularity and the other for the maximization of the intra-cluster similarity. The results obtained by these proposed metaheuristics outperformed the results from other heuristics found in the literature. However, their computational cost was high, mainly for the metaheuristic for the maximization of modularity model.

Along the years, researches revealed that the formulation that maximizes the modularity of the partitions has some limitations. In order to promote a good alternative for the maximization of the partition modularity model, this Thesis proposed new mathematical formulations for graph clustering for weighted and unweighted graphs, aiming at finding partitions with high connectivity clusters. Furthermore, the proposed formulations are able to provide partitions without a previous definition of the true number of clusters. Computational tests with hundreds of weighted graphs confirmed the efficiency of the proposed models. Comparing the partitions from all studied formulations in this Thesis, it was possible to observe that the proposed formulations presented better results, even better than the maximization of partition modularity. These results are characterized by satisfactory partitions with high correlation with the true classification for the simulated and real data (mostly biological).

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A tarefa de agrupar dados ou o problema de agrupamento de dados consiste em reunir um conjunto de objetos sem levar em consideração a sua classe ou rótulo, de forma que os objetos pertencentes a um mesmo grupo (*cluster*) apresentem forte semelhança entre si. Esse problema tem sido investigado em diversas áreas, tais como: mineração de dados (Boginski et al., 2006; Romanowski et al., 2006), alinhamento de múltiplas seqüências (Kawaji et al., 2004; Krause et al., 2005), análise de expressão gênica (Higham et al., 2007; Huttenhower et al., 2007), processamento de linguagem natural (Ushioda e Kawasaki, 1996), segmentação de imagens (Wu e Leahy, 1993) e formação de galáxias (White e Frenk, 1991). Devido à sua vasta aplicabilidade, vários algoritmos de agrupamento baseados em diferentes princípios têm sido propostos (Jain et al., 1999; Hartuv e Shamir, 2000; Hansen e Mladenović, 2001; Famili et al., 2004).

Para abordar o problema de agrupamento de dados, existem diferentes algoritmos e critérios de validação, o que conseqüentemente permite a produção de partições com grupos diversos. Por exemplo, o algoritmo de agrupamento de dados k-médias, em sua versão original, é um algoritmo que constrói partições com um determinado número de grupos definido *a priori*. Para isso, o método busca encontrar centros de grupos de forma que a partição resultante minimize a soma dos quadrados das distâncias dos objetos ao centróide (ou ponto médio) do seu *cluster*. O k-médias, devido à sua formulação, tende a formar grupos com formatos esféricos. Outro exemplo é um critério de validação bastante conhecido, o silhueta, que avalia a variância intra-*cluster*, favorecendo agrupamentos que apresentam compactação ou homogeneidade em seus grupos. A conectividade intra-*cluster*

favorece a colocação de exemplos (ou objetos) "vizinhos" em um mesmo grupo (Handl e Knowles, 2004). Logo, um mesmo algoritmo com critérios de validação ou objetivos diferentes pode gerar diferentes agrupamentos.

Os diversos algoritmos de agrupamento existentes podem ser divididos em dois grandes grupos: algoritmos hierárquicos e algoritmos particionais. Essa classificação afeta a representação da solução e a estrutura do algoritmo utilizado para a geração dos *clusters*. Nos algoritmos hierárquicos tradicionais, os *clusters* são formados gradativamente por aglomerações (algoritmos aglomerativos) ou divisões (algoritmos divisivos), gerando uma hierarquia de agrupamentos representada por meio de uma estrutura em árvore, ou seja, subgrupos ou super-grupos de agrupamentos são formados. Nos algoritmos de agrupamento particionais, o conjunto de dados é particionado em $k \leq n$ subconjuntos, em que n é o número de objetos do conjunto de dados original. A solução obtida é avaliada pelo critério de partição do algoritmo e novas soluções podem ser obtidas pela migração de objetos entre os *clusters*, até que algum critério de parada seja satisfeito.

Além das classificações tradicionais dos algoritmos de agrupamento de dados em algoritmos hierárquicos e particionais, existem outras duas classes de algoritmos que são próximas dos algoritmos particionais, denominadas de algoritmos de empacotamentos e de recobrimento (Hansen e Jaumard, 1997). Os algoritmos de empacotamento constroem $k \leq n$ subconjuntos (ou *clusters*) do conjunto de dados de forma que a intersecção entre esses subconjuntos seja vazia. A diferença desses algoritmos para os algoritmos de agrupamento particionais é que a união dos *clusters* formados por um algoritmo de empacotamento não necessariamente contém todo o conjunto de dados original, ou seja, alguns objetos do conjunto de dados original podem não pertencer a nenhum dos *clusters* da partição final. Os algoritmos de agrupamento de cobertura ou recobrimento, ao contrário dos algoritmos de empacotamento, procuram k subconjuntos (ou *clusters*) do conjunto de dados de forma que a união desses k subconjuntos seja igual ao conjunto completo de dados. Entretanto, a definição dessa classe de algoritmos permite intersecção não-vazia entre os *clusters*, ou seja, tolera *overlapping*.

Variações do problema de agrupamento de dados podem ser encontradas na literatura. Quando o conjunto de dados pode ser representado de forma eficiente em um grafo, pode-se abordá-lo como um problema de agrupamento de dados em grafo. Esse problema consiste em definir grupos de nós que apresentem alta conectividade intra-grupo e baixa conectividade inter-grupo (Kawaji et al., 2004; Krause et al., 2005; Higham et al., 2007; Huttenhower et al., 2007). Os algoritmos que têm sido propostos para resolver esse problema são basicamente algoritmos de partição do grafo, cuja função objetivo varia de acordo com as características do grafo. Exemplos de modelos de particionamento de grafos são: problema de maximização da soma das arestas intra-*cluster* (Karypis e

Kumar, 1998), problema de minimização da soma do corte entre *clusters* (Von Luxburg, 2007), problema de maximização da modularidade (Brandes et al., 2008), entre muitos outros. A modularidade é uma medida proposta por Girvan e Newman (2002) que avalia a conectividade de partições de um grafo segundo o estudo de sua configuração (mesma seqüência de graus de nós) em grafos aleatórios. Para isso, a soma das probabilidades da existência de arestas entre pares de nós intra-*cluster* é subtraída da soma dos pesos das arestas intra-*cluster*. Caso a soma das probabilidades seja alta, a existência de arestas entre pares de nós não é um evento raro, ou seja, a medida não indica uma tendência de agrupamento. Portanto, quanto maior a modularidade de uma partição, melhor a sua tendência de agrupamento.

O problema de agrupamento de dados em grafos ou problema de agrupamento em grafos também pode ser encontrado na literatura na forma de problema de detecção de comunidades em redes, denominação principalmente adotada por físicos, estatísticos e alguns cientistas da computação (Porter et al., 2009). Ambos os problemas são equivalentes, mas, normalmente, quando é mencionado como problema de comunidades, os autores procuram enfatizar o problema de redes sociais e desenvolvem estudos mais relacionados à conectividade da rede. O uso dessa última nomenclatura é conveniente, pois existe o problema de agrupamento *de* grafos[1], muitas vezes confundido com o problema aqui estudado. Essa nomenclatura é ainda mais conveniente por não haver diferenças de nomenclatura entre esses dois problemas em inglês, que são denominados *graph clustering*.

Técnicas de otimização têm sido utilizadas com sucesso para modelar problemas de agrupamento de dados, tais como os apresentados por Vinod (1969), Rao (1971) e Hansen e Mladenović (2001). O uso de modelos matemáticos para agrupamento de dados se mostrou eficiente, apresentando experiências bem sucedidas para diversos conjuntos de dados. Rao (1971) apresentou vários modelos de otimização para resolver o problema de agrupamento, incluindo modelos de programação inteira mista e não-linear. Em particular, modelos que tratam o problema de agrupamento em grafos podem ser encontrados de diferentes formas, dependendo de como se deseja agrupar os nós do grafo e do tipo de medida de conectividade utilizada para esse fim. Uma vantagem do uso de modelos é a facilidade em validar os resultados obtidos, uma vez que, por meio de sua função objetivo, é possível avaliar a qualidade da solução obtida, diferentemente de outras abordagens que necessitam de medidas específicas, como, por exemplo, aprendizado de máquina. Boas revisões que apresentam abordagens de programação matemática para problemas de agrupamento de dados podem ser encontradas em (Rao, 1971) e (Hansen e Jaumard, 1997).

Ferramentas muito utilizadas para resolver problemas de otimização de forma heu-

[1] O problema de agrupamento de grafos consiste em encontrar similaridades estruturais entre diversos grafos de forma a agrupá-los (Schaeffer, 2007; Hlaoui e Wang, 2006).

rística são as metaheurísticas. "Uma metaheurística é um conjunto de conceitos que são usados para se definir métodos heurísticos que podem ser aplicados a uma grande variedades de problemas. Em outras palavras, uma metaheurística pode ser vista como uma ferramenta algorítmica que pode ser aplicada a diferentes problemas de otimização com poucas modificações para adaptá-la a um problema específico[2]." Alguns exemplos de metaheurísticas são algoritmos genéticos (Goldberg, 1989), busca tabu (Glover e Laguna, 1997), *Greedy Randomized Search Procedures* (GRASP) (Feo e Resende, 1989), entre outros. Algumas dessas metaheurísticas têm sido amplamente usadas para a solução de problemas de agrupamento (Areibi e Vannelli, 1997; Franti et al., 1997; Lorena e Furtado, 2001). No trabalho desenvolvido nesta Tese são propostas metaheurísticas capazes de lidar de forma eficiente com problemas de agrupamento em grafos. Além disso, buscou-se enfocar o problema de agrupamento de dados em grafos do ponto de vista de otimização. Para isso, são propostos modelos de programação matemática para esse problema.

1.1 Objetivos e Resultados

O primeiro objetivo desta Tese é propor metaheurísticas para o problema de agrupamento de dados em grafos de acordo com matemáticos propostos na literatura. Os modelos para os quais são propostas metaheurísticas GRASP são:

- Maximização da similaridade (soma das arestas) intra-*cluster* (Alpert e Yao, 1994);
- Maximização da modularidade das partições (Girvan e Newman, 2002).

As metaheurísticas GRASP propostas nesta Tese apresentam diferenças em suas construções, pois suas formulações têm uma diferença fundamental: o modelo de maximização da modularidade não exige a definição prévia do número de *clusters*, definição necessária para o modelo de maximização da similaridade intra-*cluster*. Dessa forma, as metaheurísticas foram adequadas às restrições dos modelos estudados.

O segundo objetivo desta Tese é propor modelos de agrupamento baseados na conectividade dos nós de um grafo. Para isso, propõe-se uma generalização da medida *clustering coefficient* (CC), que avalia a conectividade dos *nós* de um grafo, para avaliar a tendência de agrupamento de *partições* de um grafo. Como existem duas medidas de *clustering coefficient* para avaliação de agrupamentos de nós de um grafo com pesos, um proposto por Barrat et al. (2004) e o outro por Onnela et al. (2005), duas novas medidas

[2]Definição extraída e traduzida de `http://www.metaheuristics.net` (acessado em 27/10/2010), sítio do projeto *Metaheuristics Network*, coordenado pelo Prof Dr Marco Dorigo.

são propostas, cada uma a partir de um dos pontos de partida de *clustering coefficient* de nós. A partir das medidas propostas, modelos matemáticos que as maximizem são apresentados. Para resolver heuristicamente os modelos propostos, que são caracterizados por sua não-linearidade, é proposto um algoritmo multi-nível.

Por meio de uma série de testes, que utiliza 210 grafos com tendências de agrupamento e alguns dados reais da literatura, avaliou-se a qualidade das soluções obtidas pelas metaheurísticas e modelos propostos. Com relação às metaheurísticas, resultados comparativos indicaram seu melhor desempenho com relação às outras heurísticas encontradas na literatura para os mesmos modelos. Entretanto, o tempo computacional foi mais elevado do que dos algoritmos existentes.

Os resultados obtidos para os modelos de maximização de diferentes medidas de *clustering coefficient* indicaram que um deles gerou grupos em grafos bem condizentes com os rótulos dos nós. A proposta apresentou resultados bem satisfatórios, sendo bastante competitiva com o modelo de maximização de modularidade, que vem sendo muito utilizado nos últimos anos para detectar comunidades em redes. A grande vantagem desse modelo proposto é a forma automática com que encontra os agrupamentos. Dessa forma, não é necessário definir o número de *clusters* previamente para encontrar partições usando esse critério de conectividade.

1.2 Organização da Tese

Esta Tese está dividida da seguinte maneira. No Capítulo 2 são apresentadas as principais notações adotadas, assim como conceitos de Teoria de Grafos importantes para seu entendimento. O Capítulo 3 é dedicado a uma revisão bibliográfica dos modelos de agrupamento em grafos para os quais são propostas metaheurísticas GRASP. No Capítulo 4 são descritas as metaheurísticas GRASP propostas para os modelos de maximização da soma das arestas intra-*clusters* e de maximização da modularidade de partições. No Capítulo 5 são propostas medidas de avaliação de partições baseadas no *clustering coefficient* e uma heurística multi-nível para resolver o modelo de encontrar partições que maximize essas medidas. No Capítulo 6 são apresentados e discutidos os testes computacionais realizados, onde as propostas desta Tese têm seu desempenho analisado. Por fim, no Capítulo 7 são destacados os principais resultados desta Tese e as perspectivas de trabalhos futuros.

Capítulo 2

Teoria de Grafos

O objetivo central deste capítulo é introduzir as notações que serão usadas ao longo desta Tese, assim como alguns conceitos básicos de Teoria de Grafos e de Teoria Espectral em Grafos.

2.1 Notações e Conceitos Básicos

Um grafo é definido por $G = (V(G),E(G))$, em que $V(G)$ é seu conjunto não-vazio de *vértices* (ou *nós*) e $E(G)$, seu conjunto de *arestas*. Para simplificar a notação, será adotado $V = V(G)$ e $E = E(G)$, sempre que for definido um grafo G. Caso seja necessário, ou seja, exista mais de um grafo e seja necessário distinguir os seus conjuntos de vértices e arestas, a notação inicialmente definida será utilizada.

Os elementos de V são denotados por um conjunto de números inteiros $\{1,\ldots,|V|\}$, em que $|V|$ é o número de vértices do grafo G. Cada aresta de E é definida por uma dupla (i,j), sendo que $i,j \in V$. Nesta Tese, $|V| = n$, o número de nós do grafo G, e $|E| = m$, ou seja, m é o número de arestas do grafo G. A Figura 2.1 ilustra um grafo de dois nós conectados por uma aresta.

Um grafo G pode ser definido como *direcionado* (ou *dirigido* ou *orientado*) ou *não-direcionado* (ou *não-dirigido* ou *não-orientado*) indicando, respectivamente, se existe ou não direção na relação entre dois nós, representada graficamente nas arestas por meio de setas. A Figura 2.2 ilustra um grafo direcionado em que o nó mais à esquerda é o *nó fonte*

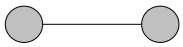


Figura 2.1: Exemplo de grafo com dois nós e uma aresta.

e o nó mais à direita é o *nó sumidouro ou terminal*. Em problemas em que um grafo é não-direcionado, pode-se simplesmente considerar os dois sentidos possíveis em todas as arestas.

Figura 2.2: Exemplo de grafo com dois nós e uma aresta direcionada.

Como o trabalho aqui desenvolvido aborda apenas grafos não-direcionados, a partir deste ponto são apenas definidos conceitos de grafos não-direcionados.

Seja $A = [a_{ij}]_{n \times n}$, em que $a_{ij} \in \{0,1\}$, a *matriz de adjacências* do grafo G. Essa matriz indica a relação de adjacência entre os nós do grafo G. Se houver uma aresta entre os nós i e j de G, então o valor de a_{ij} será 1. Caso contrário, a_{ij} será 0.

Neste trabalho, são abordados grafos com pesos em suas arestas, também chamados de *grafos ponderados* ou *grafos com pesos*. Os pesos das arestas desses grafos são representados pela *matriz de pesos* $W = [w_{ij}]_{n \times n}$, com $w_{ij} \in \mathbb{R}$. O elemento w_{ij} de W indica o peso da aresta (i,j) de G.

Outra definição utilizada nos próximos capítulos é a de *grau* dos nós de um grafo. O grau de um nó $i \in V$ é o número de arestas adjacentes a i, que é definido por meio da Equação 2.1.

$$g_i = \sum_{j=1}^{n} a_{ij} \tag{2.1}$$

Em um grafo com pesos em suas arestas, outra definição para o grau dos nós de um grafo pode ser utilizada. Essa definição é apresentada a seguir.

$$d_i = \sum_{j=1}^{n} w_{ij} \tag{2.2}$$

Na literatura, ambas as definições de grau dos nós, da Equação 2.1 e da Equação 2.2, são usadas em um grafo com pesos. Por essa razão, toda vez que for mencionado o grau

de um nó, esse será distinguido por uma das equações.

Uma *partição* de um grafo G é dada por um conjunto de grupos $C_1, C_2, \ldots, C_k$ de V, tal que, $\bigcup_{j=1}^{k} C_j = V$ e $\bigcap_{j=1}^{k} C_j = \emptyset$.

Um *caminho* é definido por uma seqüência de nós cujas arestas ligam dois vértices no grafo, levando-se em conta o sentido das arestas. Por exemplo, se um caminho ligando os vértices 1 a 3 for dado por $Seq = \{1, 2, 4, 3\}$, então a seqüência de arestas os unindo é dada por (1,2), (2,4) e (4,3). Um *caminho* é dito *simples* se nenhum de seus vértices se repete, assim como ocorre no exemplo anterior para o caminho Seq. Uma *componente conexa* é um subconjunto do grafo (*subgrafo*) que tem um caminho ligando quaisquer dois vértices pertencentes ao subgrafo. A Figura 2.3 ilustra um grafo que contém somente uma componente conexa.

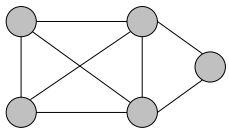


Figura 2.3: Exemplo de uma componente conexa.

O *caminho mínimo* entre dois nós de um grafo ponderado é aquele cuja soma dos pesos de suas arestas é a menor dentre todos os caminhos possíveis. Em um grafo sem pesos, a definição de caminho mínimo entre dois nós é a mesma, bastando considerar peso unitário para cada aresta do grafo. Um *ciclo* ou *circuito* em um grafo é definido por um caminho fechado, ou seja, por um caminho de G que parte de um nó i e volta a esse mesmo nó i. Diz-se que o *ciclo* é *simples* se tal caminho não tiver repetição de nós, exceto pelos nós inicial e final, que devem ser iguais. Um *grafo acíclico* é aquele que não contém ciclos.

Se houver arestas entre quaisquer dois vértices de um grafo, então se diz que ele é um grafo *completo*. Um *clique* em um grafo é um subgrafo que também é um grafo completo. O número de arestas de um clique é igual a $n_c(n_c - 1)/2$, em que n_c é o número de nós do clique. A Figura 2.4 ilustra o clique com o maior número de nós do grafo da Figura 2.3.

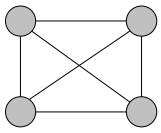


Figura 2.4: Exemplo de clique de um grafo.

Uma *árvore* é um grafo acíclico conexo. Uma *árvore geradora* é um subgrafo gerador (um subgrafo de G com todos os seus vértices) acíclico conexo. Uma *árvore geradora mínima* é uma árvore geradora com $n-1$ arestas, sendo que, no caso de grafo ponderado, a soma dos pesos de todas as arestas deve ser mínima. Uma árvore geradora mínima do grafo sem pesos da Figura 2.3 é dada na Figura 2.5.

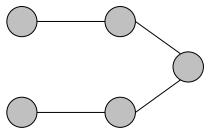


Figura 2.5: Exemplo de uma árvore geradora mínima do grafo da Figura 2.3.

O *corte* associado a um conjunto X de nós é o conjunto de todas as arestas que têm uma extremidade em X e outra em $V \backslash X$. A *contração* de arestas ou de nós advém da remoção de uma aresta do grafo e pela conseqüente junção de dois nós unidos pela aresta removida.

Existem diversos outros conceitos de Teoria de Grafos (ver Diestel (1997)), entretanto os conceitos apresentados são os mais relevantes para o entendimento dos algoritmos e métodos aqui apresentados. A seguir, são apresentados alguns conceitos de Teoria Espectral em Grafos para o entendimento de algoritmos a serem comparados nesta Tese.

2.2 Teoria Espectral em Grafos

A Teoria Espectral em Grafos iniciou-se na Química Quântica, por meio de um modelo teórico de moléculas de hidrocarbonetos não saturadas (de Abreu, 2005). Tais moléculas possuem ligações químicas com diversos níveis de energia de elétrons. Alguns desses níveis de energia podem ser representados por autovalores de um grafo, o que caracteriza o estudo de Teoria Espectral. A fundamentação teórica da Teoria Espectral em Grafos iniciou-se com Cvetković et al. (1979) e com Hall (1970), sendo popularizada nas últimas duas décadas (Chung, 1994).

O uso da teoria espectral em problemas de agrupamento em grafos é usualmente encontrado na forma de relaxação espectral de problemas de particionamento de grafos. Uma das vantagens dos algoritmos de agrupamento baseados em relaxação espectral de modelos de particionamento de grafos é a sua solução por álgebra linear padrão, que pode ser fácil de implementar por programas de computador. Vale ressaltar que os Laplacianos de um grafo, definidos pelas matrizes Laplacianas, em especial, são muito importantes

para agrupamento de dados espectral e que esta Tese se restringirá a apresentar aqueles que têm mais relevância para o tópico.

Seja W a matriz de pesos do grafo G, como definida na Seção 2.1. Seja $D = [d_{ij}]_{n\times n}$, com $d_{ij} \in \mathbb{R}$, uma matriz diagonal, definida por $d_{jj} = \sum_{i=1}^{n} w_{ij}$, com $j = 1, \ldots, n$. Vale notar que d_{jj} nada mais é do que a segunda definição apresentada nesta Tese de grau de um nó j do grafo G. A matriz Laplaciana não-normalizada de G, definida por L, é dada por:

$$L = D - W. \tag{2.3}$$

A matriz Laplaciana de um grafo sem pesos é definida por:

$$L = D - A. \tag{2.4}$$

Como os grafos estudados nesta Tese são todos ponderados, as propriedades da matriz Laplaciana L, apresentadas a seguir, serão para esse tipo de grafos. Entretanto, a equivalência dessas propriedades para a matriz Laplaciana de um grafo não ponderado é direta, bastando considerar $W = A$.

Algumas convenções e notações devem ser apresentadas antes das propriedades da matriz Laplaciana L serem expostas. A primeira se refere aos seus autovalores. Os autovalores de uma matriz quadrada M de ordem n são definidos pelas raízes do polinômio característico $p_M(\lambda) = det(M - \lambda I_n)$, em que I_n é a matriz identidade[1] de ordem n. O autovetor associado ao autovalor λ de M é dado por x tal que $Mx = \lambda x$, com $x \neq 0$ e $x \in \mathbb{R}^n$. A multiplicidade algébrica de um autovalor λ é o número de vezes que ele é raiz do polinômio característico $p_M(\lambda)$, e, para cada vez, um autovetor associado x é definido.

A matriz Laplaciana L é também conhecida como matriz de Kirchhoff (Kirchhoff, 1847), devido ao seu papel no *Matrix-Tree Theorem*. Esse teorema diz que dados dois vértices i e j de um grafo G, e considerando-se que L_{ij} é a matriz resultante da exclusão da linha i e da coluna j da matriz L, o valor absoluto do determinante de L_{ij} é igual ao número de árvores geradoras do grafo G. Uma demonstração desse teorema pode ser encontrada em (Mohar, 1991).

Os teoremas e propriedades a seguir foram extraídos de (Mohar, 1991), (Hagen e Kahng, 1992) e (Von Luxburg, 2007).

Proposição 2.2.1. *A matriz L satisfaz as seguintes propriedades:*

[1] A matriz identidade é a matriz diagonal cujos elementos da diagonal são 1.

(i) Para cada vetor $x \in \mathbb{R}^n$, tem-se que $x^t L x = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2$.

(ii) L é uma matriz simétrica e semi-definida positiva.

(iii) L tem n autovalores não negativos.

(iv) O menor autovalor de L é $\lambda_1 = 0$, com o autovetor correspondente $\alpha\mathbb{1}$, $\alpha \in \mathbb{R}$, e $\mathbb{1}$ é o vetor unitário de dimensão n.

Para provar (i), considere as seguintes igualdades.

$$x^t L x = x^t D x - x^t W x = \sum_i d_{ii} x_i^2 - \sum_{i,j} w_{ij} x_i x_j = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2.$$

Por (i), tem-se que a matriz Laplaciana L é semi-definida positiva, pois $x^t L x \geq 0$ desde que os pesos do grafo sejam todos positivos. Além disso, a matriz L é simétrica, pois $L = D - W = D^t - W^t$, provando a propriedade (ii). A propriedade (iii) é conseqüência do fato da matriz L ser simétrica e semi-definida positiva. A propriedade (iv) pode ser deduzida facilmente, pois, por (iii), tem-se que todos os autovalores de L são maiores ou igual a zero. Além disso,

$$\begin{bmatrix} \sum_{i\neq 1}^{n} w_{i1} & -w_{12} & \cdots & -w_{1n} \\ -w_{21} & \sum_{i\neq 2}^{n} w_{i2} & \cdots & -w_{1n} \\ \vdots & & \ddots & \vdots \\ -w_{n1} & -w_{n2} & \cdots & \sum_{i\neq n}^{n} w_{in} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} \sum_{i\neq 1}^{n} w_{i1} - \sum_{i\neq 1}^{n} w_{i1} \\ \sum_{i\neq 2}^{n} w_{i2} - \sum_{i\neq 2}^{n} w_{i2} \\ \vdots \\ \sum_{i\neq n}^{n} w_{in} - \sum_{i\neq n}^{n} w_{in} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Ou seja, $L\alpha\mathbb{1} = 0\alpha\mathbb{1}$, o que implica que 0 é autovalor de L.

A proposição seguinte diz respeito ao número de componentes conexas de G e ao espectro de L.

Proposição 2.2.2. *Seja G um grafo não direcionado com pesos não negativos. A multiplicidade k do autovalor 0 de L corresponde ao número de componentes conexas de G. Considere $C_1, \ldots, C_k$ tais componentes conexas. O auto-espaço do autovalor 0 é gerado pelos vetores indicadores $\mathbb{1}_{C_1}, \ldots, \mathbb{1}_{C_k}$.*

Na Proposição 2.2.2, $\mathbb{1}_{C_i}$ é o vetor com 1 nas posições correspondentes à ordem dos vértices pertencentes à componente conexa C_i, ou seja, se o vértice j pertencer à componente conexa C_i, então a j-ésima posição do vetor $\mathbb{1}_{C_i}$ terá valor 1, caso contrário, será 0. Uma conseqüência da Proposição 2.2.2 é a propriedade encontrada em Boccaletti et

al. (2006), que diz que se o grafo apresentar uma estrutura com k *clusters* aparentes (não necessariamente perfeita), então haverá apenas um autovalor igual a 0 e $k-1$ autovalores bastante próximos de 0, enquanto os autovalores restantes serão bem maiores do que 0.

Como visto, os autovalores podem expressar algumas propriedades de componentes conexas de um grafo. Um autovalor importante que apresenta relações entre componentes conexas é o λ_2. Fiedler (1975) sugeriu que o autovetor associado ao autovalor λ_2, que mais tarde veio a ser denominado autovetor de Fiedler, poderia ser utilizado para encontrar uma aproximação para o problema de particionamento do grafo em dois conjuntos. Tal aproximação é feita com base nos sinais dos componentes do autovetor de Fiedler. O autor apresentou uma expressão para esse autovalor e denominou o seu valor de *conectividade algébrica* do grafo G. Para mais detalhes, ver (Fiedler, 1975).

Diferente da formulação de Fiedler, a seguinte equação para o λ_2 é comumente usada:

$$\lambda_2(G) = \min_{x \perp \mathbb{1}, x \neq \mathbb{0}} \frac{x^t L x}{x^t x} = \min_{x \perp \mathbb{1}, x \neq \mathbb{0}} \frac{1}{2} \frac{\sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2}{||x||^2}. \tag{2.5}$$

Esses são os principais conceitos para o entendimento dos algoritmos espectrais de agrupamento de dados apresentados nesta Tese. As notações apresentadas neste capítulo são adotadas nos capítulos posteriores, sem a repetição de sua definição.

A seguir, é apresentada uma revisão bibliográfica de problemas de particionamento que são estudados nesta Tese.

Capítulo 3

Modelos de Agrupamento em Grafos

O problema de agrupamento de dados em grafos vem sendo explorado há várias décadas, utilizando diversas abordagens. Algumas revisões desse tema, tais como (Cormack, 1971), (Hansen e Jaumard, 1997), (Schaeffer, 2007) e (Filippone et al., 2008), possibilitam uma visão geral do que tem sido desenvolvido nessa área envolvendo otimização. Em particular, Schaeffer (2007) apresenta uma revisão do problema de agrupamento em grafos mostrando a grande quantidade de publicações e diferentes abordagens nesse tema. Algumas dessas abordagens são baseadas na conectividade dos nós do grafo, na homogeneidade e na densidade dos nós, o que permite, conseqüentemente, gerar *clusters* com diversas características diferentes.

Segundo Beliakov e King (2006), devido à falta de uma formulação matemática precisa para definir um conjunto de conceitos distintos para análise de agrupamento, um estudo formal de suas metodologias não é usualmente realizado. A Pesquisa Operacional pode promover um formalismo matemático, assim como prover ferramentas importantes para modelos de agrupamento. Além disso, a programação matemática pode fornecer índices de validação das soluções obtidas. Os modelos matemáticos existentes para agrupamento em grafos são diversos, havendo diferentes objetivos para o particionamento de grafos. Esses diversos modelos são capazes de produzir *clusters* de diferentes formatos que podem ser mais adequados para determinadas conformações dos dados.

Existem vários modelos e algoritmos para agrupamento em grafos. Esta revisão bibliográfica tenta abranger os algoritmos de particionamento de grafos que são usados para abordar o problema de agrupamento em grafos usando dois critérios: maximização

da similaridade intra-*cluster* e maximização da modularidade das partições. Entretanto, alguns outros modelos de particionamento atualmente estudados para o problema de agrupamento em grafos são apresentados, sem uma revisão muito aprofundada da literatura em que se discuta os diversos algoritmos existentes para eles, para não se perder o foco do estudo desta Tese.

3.1 Formulações de Corte e Particionamento de Grafos

Como já foi mencionado anteriormente, existem diferentes abordagens na literatura para tratar o problema de particionamento de grafos. A diversidade de formulações é originada das limitações e das vantagens que cada uma apresenta. Algumas dessas formulações são apresentadas a seguir.

Os problemas de particionamento de grafos em k *clusters* podem apresentar diferentes nomenclaturas quando deseja-se particionar o grafo em dois (bi-particionamento) ou em mais grupos (k-particionamento). Formalmente, o problema de k-particionamento baseado em formulações de cortes de grafos tem como objetivo eliminar arestas de um dado grafo de forma a produzir k componentes conexas. Geralmente, o objetivo das formulações para esse problema é fornecer *clusters* bem separados com alta conectividade intra-*cluster*. Para tal, a maioria desses critérios determina diferentes alternativas para a escolha do conjunto de arestas a serem eliminadas. Considere a seguinte formulação:

$$corte(\pi^k) = \frac{1}{2}\sum_{i=1}^{k} W(C_i,\bar{C}_i). \tag{3.1}$$

em que $W(C_r,C_t) = \sum_{\substack{i\in C_r \\ j\in C_t}} w_{ij}$; $\bar{C}_i$ é todo C_j tal que $j \neq i$; π^k é uma k-partição[1]; e Π^k o conjunto de todas as k-partições de G.

Problema do Corte Mínimo

O primeiro problema de particionamento a ser apresentado é o de minimização do corte da k-partição. Ele consiste na seleção do conjunto das arestas cuja soma dos pesos é a menor possível. Tal formulação é apresentada na Equação 3.2.

$$\min_{\pi^k\in\Pi} corte(\pi^k) \tag{3.2}$$

[1]Uma k-partição é um partição do grafo G com k *clusters*.

Essa formulação tem o objetivo de minimizar a soma dos pesos das arestas entre os pares de nós de diferentes *clusters*, $corte(\pi^k)$. Wu e Leahy (1993) observaram que as partições, dependendo das características do grafo a ser particionado, podem ter *clusters* com um único nó, o que pode não ser desejável, como por exemplo, em planejamento de circuitos. Por esse motivo, outras formulações que levam em consideração aspectos adicionais foram propostas na tentativa de resolver esse problema. Um exemplo é encontrado em (Alpert et al., 1999), em que os autores consideraram limitantes inferiores e superiores para delimitar o tamanho dos *clusters* da partição π^k. Nesse caso, um conjunto de restrições é adicionado à formulação (3.2): $\mathcal{L}_i \leq |C_i| \leq \mathcal{U}_i$, para $1 \leq i \leq k$. Os parâmetros $\mathcal{L}_i$ e $\mathcal{U}_i$ são, respectivamente, os limitantes inferiores e superiores do i-ésimo *cluster* e $|C_i|$ é o número de nós de um *cluster* C_i.

Problema de Minimização do Raio do Corte

Outra abordagem para evitar o problema de partições com nós isolados em um único *cluster* é considerar a Equação 3.2 dividida pelo número de nós em cada *cluster*. Essa formulação foi inicialmente proposta para o problema de bi-particionamento (Leighton e Rao, 1988; Wei e Cheng, 1989), e é conhecida como problema de minimização do raio da bipartição. Anos depois, ela foi generalizada por Chan et al. (1994) para o problema de minimização do raio do corte da k-partição, por meio de sua conexão com o problema de alocação quadrática (Hall, 1970). Essa nova formulação é representada pela Equação 3.3.

$$\min_{\pi^k \in \Pi^k} \text{raio corte}(\pi^k) \tag{3.3}$$

em que:

$$\text{raio corte}(\pi^k) = \frac{1}{2} \sum_{i=1}^{k} \frac{W(C_i,\bar{C}_i)}{|C_i|}. \tag{3.4}$$

Os algoritmos encontrados na literatura para a solução desse problema são, em sua maioria, algoritmos espectrais (Hagen e Kahng, 1992; Von Luxburg, 2007). Para mais detalhes desses algoritmos, recomenda-se a leitura de (Von Luxburg, 2007).

Problema de Minimização do Corte Normalizado

Outro modelo que tem sido investigado para agrupamento em grafos é o problema de minimização do k-corte normalizado. Nesse problema, cada parcela do corte das arestas, $W(C_i,\bar{C}_i)$, é fracionado pela soma dos graus dos nós dentro do *cluster* C_i, que é dado por $vol(C_i) = \sum_{j \in C_i} d_{jj}$. A formulação desse problema é apresentada na Equação 3.5.

$$\min_{\pi^k \in \Pi^k} ncut(\pi^k) \tag{3.5}$$

em que:

$$ncut(\pi^k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(C_i,\bar{C}_i)}{vol(C_i)}. \tag{3.6}$$

Esse problema foi proposto por Shi e Malik (1997, 2000), que o derivaram de uma relação entre medidas normalizadas de associação e de desassociação de nós de um grafo. A associação normalizada reflete a média da conectividade dos nós dentro do *cluster*. A medida de desassociação mede o custo do corte como uma porcentagem do corte das arestas com relação a todos os nós do grafo. Os autores concluíram que o problema de encontrar uma partição que minimize a desassociação entre *clusters* é o mesmo que encontrar uma partição que maximize a associação dentro dos *clusters*. Os mesmos autores propuseram o uso de algoritmos espectrais para resolver esse problema.

Problema de Minimização do Corte Min-max

Essa formulação, o problema de minimização do corte min-max, foi proposta por Ding et al. (2001, 2004) e tem como objetivo a minimização das somas das similaridades inter-*clusters* e a simultânea maximização das somas das similaridades intra-*cluster*. Ela pode ser expressa pela Equação 3.7.

$$\min_{\pi^k \in \Pi^k} \text{corte } min\text{-}max(\pi^k) \tag{3.7}$$

em que:

$$\text{corte } min\text{-}max(\pi^k) = \sum_{i=1}^{k} \frac{W(C_i,\bar{C}_i)}{W(C_i,C_i)}. \tag{3.8}$$

Novamente, os próprios autores que propuseram essa formulação desenvolveram uma relaxação espectral da mesma para a posterior proposição de uma heurística para resolvê-la.

Esses modelos de cortes foram estudados e, como resultado, decidiu-se por abordar o problema de minimização do corte de arestas, que também pode ser abordado como o problema de maximização da soma das similaridades intra-*cluster*. O primeiro motivo de utilizá-lo é o fato de ele ser mais intuitivo do que os outros modelos de corte. O segundo é que nenhum dos outros modelos garantiu resultados muito melhores do que o de minimização do corte, que é mais simples. O terceiro é que os autores criticam o modelo de mínimo corte por produzir *clusters* com números de nós diferentes, ou seja, partição

não balanceada, que pode ser interessante em algumas aplicações de agrupamento em grafos.

Nos testes computacionais, o problema de apresentar *clusters* com nós isolados foi observado. Entretanto, as partições não foram apenas caracterizadas por essa limitação. Pelo contrário, essa formulação se apresentou eficiente, e com partições próximas de classificações verdadeiras de bases biológicas, quando o grafo estudado era completo.

3.2 Formulação Matemática para a Maximização da Soma das Arestas Intra-Cluster

Esse modelo procura encontrar uma partição em um grafo de forma a maximizar o número de arestas compartilhadas dentro dos *clusters*. Se o grafo for ponderado, o modelo visará a maximização dos pesos das arestas intra-*cluster*. Um modelo matemático não-linear para esse problema é apresentado a seguir. Ele tem grande semelhança com o problema de designação quadrática e pode ser encontrado em (Rao, 1971) e (Kochenberger et al., 2005) para problemas de agrupamento de dados.

$$max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} \sum_{t=1}^{k} x_{it} x_{jt}$$

sujeito a:

$$\sum_{j=1}^{k} x_{it} = 1 \qquad i = 1 \ldots n \tag{3.9}$$

$$\sum_{i=1}^{n} x_{it} \geq 1 \qquad t = 1 \ldots k \tag{3.10}$$

$$x_{it} \in \{0,1\} \qquad i = 1 \ldots n, t = 1 \ldots k \tag{3.11}$$

em que:

x_{it} é uma variável binária que assume valor 1 se o nó i pertence ao *cluster* t e 0, caso contrário.

As Restrições (3.9) forçam cada nó i a pertencer a um *cluster*. As Restrições (3.10) garantem que o *cluster* j tenha pelo menos um nó contido nele. Por fim, as Restrições (3.11) definem as variáveis x_{it} como binárias.

Esse modelo é caracterizado por ter uma função objetivo não-linear, logo, é necessário usar técnicas de programação inteira não-linear para resolvê-lo. Nesse caso, não há nenhum algoritmo polinomial capaz de resolvê-lo na otimalidade. Um modelo linear equivalente ao anterior é apresentado a seguir.

$$max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} y_{ij}$$

sujeito a:

$$(3.9) - (3.11)$$

$$y_{ij} \geq x_{it} + x_{jt} - 1 \qquad i = 1 \ldots n, j = i+1 \ldots n, t = 1 \ldots k \tag{3.12}$$

$$y_{ij} \in \{0,1\} \qquad i = 1 \ldots n, j = i+1 \ldots n \tag{3.13}$$

em que:

y_{ij} é uma variável binária que assume valor 1 se os nós i e j pertencerem ao mesmo *cluster*, e 0, caso contrário.

As Restrições (3.12) e (3.13) garantem que y_{ij} assuma o valor 1 se ambas as variáveis x_{it} e x_{jt} forem iguais a 1. Esse modelo linear tem $n^2/2 - n$ variáveis a mais do que o modelo inteiro anteriormente apresentado e $n(n-1)(k+1)/2$ restrições adicionais. Para resolvê-lo na otimalidade pode-se utilizar um pacote de otimização, como o CPLEX ILOG (2007). O CPLEX é um pacote que utiliza a técnica *branch and bound* and *cut* e muitos outros operadores de pré-processamento para tratar problemas de programação inteira mista. Entretanto, a utilização desses pacotes não é viável em problemas grandes, com um número de nós maior do que 150, como foi apresentado por Nascimento et al. (2010b). Por essa razão, o uso de heurísticas eficientes se mostra necessário para encontrar uma solução de boa qualidade para o problema.

As heurísticas encontradas na literatura para resolver de forma heurística o problema de particionamento de grafos que minimizem o corte ou maximizem a soma dos pesos das arestas intra-*cluster* são diversas, como por exemplo, as descritas em (Alpert

e Yao, 1994), (Karypis e Kumar, 1996) e (Alpert et al., 1999). Alpert e Yao (1994) e Alpert et al. (1999) reduziram o problema de minimização de cortes de um grafo ao problema de particionamento de um vetor. Esse problema consiste em encontrar um partição $P^k = \{P_1, \ldots, P_k\}$ de um conjunto de vetores Y. Limitantes superiores e inferiores para delimitar o tamanho dos conjuntos P_i com $1 \leq i \leq k$ podem ser definidos. Para resolver heuristicamente o problema discutido, os autores propuseram o uso de um algoritmo guloso de ordenação linear no problema reduzido. Nessa redução, o conjunto de vetores a serem particionados é calculado usando o conjunto dos $d \geq k$ autovetores associados aos d menores autovalores da matriz Laplaciana L do grafo a ser particionado. Depois de ordenado, o vetor é particionado por meio do algoritmo de programação dinâmica de Alpert e Kahng (1994). Por usar os autovalores da matriz Laplaciana do grafo, esse algoritmo faz parte da classe de algoritmos espectrais.

Seguindo essa mesma linha de algoritmos, mais recentemente, outras abordagens mais simples para encontrar uma partição do conjunto dos autovetores associados à matriz Laplaciana do grafo a ser particionado foram propostas. Para o modelo de particionamento estudado, existe, por exemplo, o algoritmo apresentado em (Von Luxburg, 2007), conhecido como *Unnormalized k-means Algorithm* (Ukmeans). Esse algoritmo gera uma solução para o modelo abordado nesta seção depois de uma relaxação do mesmo. A matriz dos k autovetores associados aos k menores autovalores dispostos na coluna da matriz é a solução do modelo relaxado. A heurística proposta por Von Luxburg (2007) para encontrar uma solução factível para o problema original consiste na aplicação do algoritmo k-médias[2] (Kaufman e Rousseeuw, 1990) à matriz de autovetores. O resultado é encontrado considerando a i-ésima linha da matriz representando o nó i do grafo a ser particionado.

Outra classe de algoritmos que foram propostos para resolver esse problema são os algoritmos multi-níveis. Eles representam uma categoria de algoritmos de particionamento que têm se mostrado eficientes devido à sua robustez e estabilidade. Alguns exemplos desses algoritmos em grafo podem ser encontrados em (Barnard e Simon, 1994), (Hendrickson e Leland, 1995), (Karypis e Kumar, 1996) e (Karypis e Kumar, 1998). Esses algoritmos são compostos basicamente por três fases, que são descritas a seguir.

1. **Fase de contração**: Nessa fase, o grafo é sucessivamente comprimido (contração dos nós) em alguns passos, até atingir a fase de particionamento inicial, onde ele está, segundo o critério do algoritmo de contração, na sua condição base.

2. **Fase de particionamento inicial**. O grafo base é particionado segundo algum

[2]O algoritmo k-médias consiste em minimizar o soma dos quadrados das distâncias entre os objetos e o centróide do *cluster* ao qual eles pertencem.

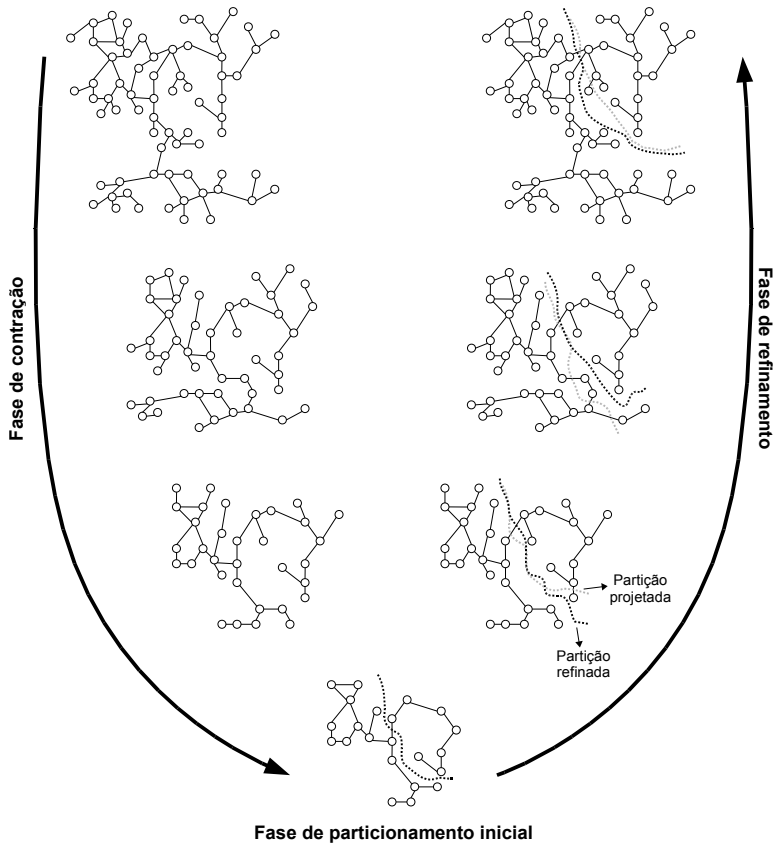


Figura 3.1: Ilustração de particionamento por algoritmo multi-nível de um grafo.

algoritmo de particionamento.

3. **Fase de refinamento**. O grafo base é sucessivamente expandido até sua estrutura original. A partição da base inicial é melhorada durante o processo de refinamento.

Um esquema que ilustra um típico algoritmo multi-nível é apresentado na Figura 3.1.

Segundo Karypis e Kumar (1996), os algoritmos multi-níveis são rápidos se comparados, por exemplo, a algoritmos espectrais. São capazes de particionar grafos com milhares de nós e arestas em poucos segundos. Além disso, segundo os autores, são algoritmos paralelizáveis.

Diversos algoritmos multi-níveis foram propostos para o problema discutido nesta seção. Alguns desses algoritmos têm como restrição adicional gerar partições cujos *clusters* tenham tamanhos semelhantes. Barnard e Simon (1994) propuseram um algoritmo multi-nível recursivo espectral para o problema de bi-particionamento de grafos. A fase de contração desse algoritmo é feita por meio de emparelhamentos maximais, estratégia comumente empregada nessa fase dos algoritmos multi-níveis. Um emparelhamento em

um grafo G é definido por um conjunto S de arestas em que, para cada $(i,j) \in S$, não há nenhuma aresta (k,j), com $k \neq i$ ou (k,i) com $k \neq j$ que pertença a S. Um emparelhamento S é dito maximal se, para qualquer aresta do grafo que não esteja no emparelhamento, exista ao menos uma de suas pontas[3] em alguma aresta do conjunto S. Cada aresta do conjunto maximal é contraída e os pesos das arestas adjacentes a esses nós são recalculados. Para a fase de refinamento, Barnard e Simon (1994) apresentaram um algoritmo iterativo baseado no quociente de Raleigh, aproximando o valor do autovetor de Fiedler.

Hendrickson e Leland (1995) executaram em seu trabalho a contração por emparelhamentos maximais. Para isso, eles propuseram uma generalização para k-partições de um algoritmo encontrado na literatura para o problema de bisseção de grafo para compor a fase de particionamento da base inicial. Na fase de refinamento, os autores utilizaram uma versão da busca local de Kernighan e Lin (1970).

Karypis e Kumar (1998) usaram duas abordagens para a contração dos nós, uma encontrando um emparelhamento maximal aleatório e a posterior contração dos nós do emparelhamento, e a outra pela criação de multi-nós de vértices altamente conexos. Com relação à fase de particionamento da base inicial, eles utilizaram quatro algoritmos diferentes de k partições: um de bi-particionamento espectral generalizado e os outros três de natureza combinatorial, que tentam produzir bisseções com cortes de arestas de pesos baixos utilizando várias heurísticas. A fase de refinamento de Karypis e Kumar (1998) envolve a projeção e o refinamento das partições que retornam ao espaço original. A cada passo, é realizada uma busca local. Para realizar a busca local, os autores também usaram a heurística proposta por Kernighan e Lin (1970).

Mais recentemente, Dhillon et al. (2007) propuseram um algoritmo multi-nível para agrupamento de dados semelhante ao proposto por Karypis e Kumar (1998). A fase de contração é caracterizada por um procedimento de corte mínimo. Essa fase, segundo os autores, permite que outros objetivos para particionamento do grafo sejam abordados. Para particionar a base inicial, os autores propuseram um método de bi-particionamento generalizado para k partições. Os autores também avaliaram outros dois métodos de particionamento, que são: um algoritmo espectral (Yu e Shi, 2003) e um algoritmo de crescimento de vizinhança, que agrupa o grafo base selecionando os vértices aleatoriamente e expandindo a região em torno desses vértices pela estratégia de busca em largura. A fase de refinamento é caracterizada pelo uso de um algoritmo baseado no k-médias e uma operação de busca local, em que há transferência de nós entre os *clusters*.

[3]Uma ponta em uma aresta é um dos nós conectados por essa aresta.

3.3 Formulação Matemática para a Maximização da Modularidade

Conforme discutido no Capítulo 1, encontrar padrões em dados é um problema de ampla aplicabilidade em diversos ramos da ciência, e uma maneira de identificar padrões em um conjunto de dados é dividi-lo em *clusters* para posterior análise. Citando um exemplo, há diversas décadas o problema de detecção de comunidades vem sendo estudado. Esse problema consiste em determinar comunidades em uma rede social para se explicar o comportamento de grupos de indivíduos. Estudos datados da década de 20 sobre esse problema podem ser encontrados, como é o caso de (Rice, 1927), em que o autor realiza uma detecção manual de grupos políticos.

Uma maneira de se representar as redes sociais é por meio de grafos. Para isso, basta considerar os nós como os indivíduos e suas interações pelas arestas conectando os nós. Essas interações podem ter intensidade, sendo representadas pelos pesos das arestas entre os nós dos grafos. Dessa forma, o problema de encontrar comunidades sociais se reduz ao problema de encontrar agrupamentos em grafos, ou ao problema de particionamento de grafos ou, ainda, ao problema de detecção de comunidades em redes[4].

O problema de detecção de comunidades em redes consiste em identificar grupos de nós, que também podem ser chamados de *módulos*, com alta conectividade intra-grupos. Recentemente, um avanço envolvendo o problema de detecção de comunidades feito por Girvan e Newman (2002) despertou a atenção de físicos e matemáticos. Nesse trabalho, os autores apresentam uma nova medida de avaliação de comunidades, que pode ser otimizada e utilizada na detecção de comunidades. Essa medida, conhecida como *modularidade*, tem raízes na mecânica estatística. Grosso modo, ela consiste na avaliação da conectividade de partições de uma dada rede tendo em vista a sua mesma configuração (mesma seqüência de graus de nós) em grafos aleatórios. Se a tendência de agrupamento desses nós não for um evento comum, então a medida é favorecida, desde que haja um indício de agrupamento da partição. Essa medida é mais bem detalhada a seguir.

3.3.1 Modularidade

A modularidade é uma medida proposta nesta década para a avaliação de qualidade de algoritmos de agrupamento em grafos (Girvan e Newman, 2002). A seguinte formulação é considerada como ponto de partida para sua definição:

[4]Neste trabalho, redes têm a mesma definição de grafo.

$$q(\pi) = \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} a_{ij}\delta(c_i,c_j)}{2m}, \tag{3.14}$$

em que π é uma partição do grafo G com k *clusters*; c_i é o *cluster* ao qual o nó i pertence; e $\delta(c_i,c_j)$ é uma função que recebe o valor 1 se os nós i e j pertencem ao mesmo *cluster*, e 0, caso contrário.

O numerador de $q(\pi)$ indica a soma do número de arestas intra-*cluster*, somada duas vezes, por isso a divisão por 2. Como m é o número total de arestas do grafo, $q(\pi)$ avalia o número de conexões dentro dos *clusters* com relação ao grafo original. Logo, quanto mais próximo de 1 o valor de $q(\pi)$ estiver, melhor é a conectividade da partição.

Entretanto, a formulação 3.14 é considerada fraca, pois uma partição de apenas um *cluster*, correspondendo a todo o grafo, terá modularidade 1. Para contornar esse problema, uma outra formulação para a modularidade é considerada. Essa formulação é apresentada na Equação 3.15 e visa medir a diferença entre o número de arestas unindo os vértices do mesmo *cluster* e o número esperado dessas arestas em um grafo aleatório.

$$q(\pi) = \frac{1}{2m} \sum_{i=1}^{k} \sum_{j=1}^{k} \left[a_{ij} - \frac{g_i g_j}{2m} \right] \delta(c_i,c_j) \tag{3.15}$$

A parcela $\frac{g_i g_j}{2m}$ indica a probabilidade de existir uma aresta (i,j) em um grafo aleatório mantendo o grau dos nós do grafo original, lembrando que g_i é o grau do nó i definido pela Equação 2.1. Nessa medida, o pior caso ocorre quando o grafo tiver modularidade 0 e o grafo aleatório tiver modularidade 1, ou seja, quando $q(\pi) = -1$. O melhor caso ocorre quando o grafo tiver modularidade 1 e o grafo aleatório tiver modularidade 0, ou seja, quando $q(\pi) = 1$.

Se o grafo tiver pesos, a definição de modularidade de sua partição é dada por:

$$q(\pi) = \frac{1}{2m} \sum_{i=1}^{k} \sum_{j=1}^{k} \left[w_{ij} - \frac{d_i d_j}{2m'} \right] \delta(c_i,c_j) \tag{3.16}$$

em que d_i é a definição de grau de um nó dada pela Equação 2.2 e $m' = \sum_{i=1}^{n} d_i/2$.

Diversas propriedades dessa medida de qualidade e caracterizações do grafo segundo essa medida foram estudadas por Brandes et al. (2008). Adicionalmente, os autores desenvolveram um modelo linear inteiro que fornece uma partição que maximiza a modularidade de um grafo G, dado por:

$$maximize \sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}x_{ij}$$

sujeito a:

$$x_{ij} + x_{jk} - 2x_{ik} \leq 1 \qquad 1 \leq i,j,k \leq n \tag{3.17}$$

$$x_{ij} + x_{ik} - 2x_{jk} \leq 1 \qquad 1 \leq i,j,k \leq n \tag{3.18}$$

$$x_{ik} + x_{jk} - 2x_{ij} \leq 1 \qquad 1 \leq i,j,k \leq n \tag{3.19}$$

em que $r_{ij} = \frac{1}{2m'}(w_{ij} - \frac{d_i d_j}{2m'})$ se o grafo tiver pesos e $r_{ij} = \frac{1}{2m}(a_{ij} - \frac{g_i g_j}{2m})$, caso o grafo não tenha pesos em suas arestas. Nessas equações, x_{ij} é um variável binária que assume o valor 1 se os nós i e j pertencem ao mesmo *cluster*, e 0, caso contrário.

Para resolver o problema de detecção de comunidades, diversas heurísticas foram propostas. Algumas delas são de arredondamento, ou seja, a heurística arredonda a solução do problema relaxado linearmente (Agarwal e Kempe, 2008); heurísticas baseadas na relaxação espectral do modelo original, cuja solução é dada pelos autovetores da matriz $R = [r_{ij}]_{n\times n}$ (Newman, 2006); algoritmo multi-nível (Noack e Rotta, 2009), entre outras. Outra heurística para esse problema foi proposta por Clauset et al. (2004). Ela consiste em um procedimento guloso seguido de uma busca local. A busca local empregada nesse algoritmo é a de Kernighan e Lin (1970), que permite movimentos que originam soluções piores do que a solução corrente da iteração.

A seguir, serão discutidas as técnicas de validação adotadas nesta Tese.

3.4 Técnicas de Validação

Agrupamento de dados tem se mostrado uma subárea de aprendizado de máquina bem madura e consolidada. Entretanto, seu estudo em grafos é ainda uma área amplamente inexplorada. Logo, validar resultados de algoritmos de agrupamento de dados em grafos é um grande desafio. Apesar disso, algumas medidas e índices para verificar a eficiência dos algoritmos de agrupamento são usados como critérios de validação. Mesmo não sendo totalmente apropriadas, elas analisam a estrutura obtida para verificar se o agrupamento satisfaz requisitos mínimos de estabilidade e acurácia. Esses critérios associam um valor a um agrupamento ou partição, constituindo assim uma importante ferramenta para avaliar a qualidade. Esses critérios permitem verificar quão adequadas são as partições produzidas para um determinado conjunto de dados. Vários critérios que avaliam diferentes aspectos

das partições geradas foram publicados, tais como (Rezaee et al., 1998), (Theodoridis e Koutroubas, 1999), (Xie e Beni, 1991) e (Pal e Biswas, 1997).

Segundo Theodoridis e Koutroubas (1999), três grandes abordagens têm sido seguidas para a validação de *clusters*: critérios internos, que avaliam os resultados de um dado algoritmo de agrupamento segundo os valores do vetor de atributos, por exemplo, a matriz de similaridades (análise mais quantitativa); critérios externos, que avaliam o algoritmo de agrupamento de acordo com a intuição ou conhecimento sobre a estrutura verdadeira do conjunto de dados; e, por fim, os critérios relativos, que empregam uma avaliação mais robusta cuja idéia básica é avaliar a estrutura da partição comparando os resultados de um dado algoritmo de agrupamento, usando o próprio algoritmo com valores de parâmetros diferentes. Os critérios estudados e adotados neste trabalho são do tipo interno e externo. O critério de avaliação interna de cada modelo é avaliado por sua própria função objetivo. Já o critério externo pede uma definição mais detalhada, a ser fornecida no próximo parágrafo.

Um dos índices externos mais comumente usados é o índice *Rand* corrigido, chamado aqui de *CRand* (Hubert e Arabie, 1985). Esse índice será usado para avaliar as partições dos algoritmos propostos nesta Tese. Ele avalia as partições com relação às classificações reais fornecidas nos conjuntos de dados. O índice *Rand* avalia a similaridade entre duas partições por meio de uma análise de concordância e discordância entre pares de objetos das partições. O índice *Rand* considera dois objetos concordantes em duas partições se, em ambas, os objetos pertencem ao mesmo *cluster* ou se, em ambas as partições, os objetos pertencem a *clusters* diferentes. O índice *CRand* é uma normalização do *Rand* e é dado pela equação a seguir.

$$CRand = \frac{\sum_{i=1}^{K^a}\sum_{j=1}^{K^b}\binom{|C_i^a \cap C_j^b|}{2} - \left[\sum_{i=1}^{K^a}\binom{|C_i^a|}{2}\sum_{j=1}^{K^b}\binom{|C_j^b|}{2}\right]/\binom{n}{2}}{\left[\sum_{i=1}^{K^a}\binom{|C_i^a|}{2} + \sum_{i=1}^{K^b}\binom{|C_i^b|}{2}\right]/2 - \left[\sum_{i=1}^{K^a}\binom{|C_i^a|}{2} + \sum_{i=1}^{K^b}\binom{|C_i^b|}{2}\right]/\binom{n}{2}} \tag{3.20}$$

em que π^a e π^b são as duas partições a serem comparadas, K^a e K^b, o número de *clusters* das partições π^a e π^b, respectivamente, e C_i^a e C_j^b, o *cluster* i de π^a e o *cluster* j de π^b, respectivamente. O valor de *CRand* varia no intervalo $[-1,1]$. Quanto mais próximo de 1 esse valor estiver, mais concordantes são as duas partições.

Pode-se observar que existem diversos índices e medidas para avaliar os agrupamentos e que muitos deles podem ser conflitantes. Por essa razão, é usado apenas esse índice

externo para que a avaliação seja mais estável e direcionada ao propósito desta Tese.

3.5 Considerações Finais

Os algoritmos de agrupamento de dados em grafos são algoritmos que buscam particionar um grafo em grupos de nós com alta conectividade intra-*cluster*. A forma de se definir tal conectividade pode ser traduzida de diversas maneiras, que podem ser tratadas como função objetivo do problema de particionamento. Este capítulo apresentou algumas das formulações que são mais estudadas para particionamento de grafos, sendo que para duas delas são propostas metaheurísticas nesta Tese. Além disso, foi apresentada uma revisão bibliográfica das heurísticas encontradas na literatura para essas duas formulações: a de maximização da soma das arestas intra-*cluster* e a de maximização da modularidade das partições.

O modelo de maximização da similaridade intra-*cluster* não vem sendo muito estudado nos últimos anos para agrupamento de dados, sob a justificativa de que ele produz partições insatisfatórias. Para minimizar tais limitações, outras funções objetivo foram propostas, com a intenção de produzir agrupamentos de melhor qualidade. Entretanto, por meio de testes realizados nesta Tese e em (Nascimento et al., 2010b), existem casos em que esse modelo apresenta resultados muitos bons.

Em contraste, o modelo de maximização da modularidade tem sido bastante explorado desde que foi proposto. Algumas limitações desse modelo foram apresentadas por Porter et al. (2009). Entretanto, os autores apontam que esse modelo precisa ser bem explorado de forma a se estudar melhor os agrupamentos por ele produzidos. Uma maneira de realizar esse estudo é a proposta de heurísticas que produzam partições mais próximas da solução ótima, para se analisar adequadamente as características dos agrupamentos em diversos grafos. Esse é o objetivo desta Tese: propor heurísticas que cumpram com seu objetivo de encontrar soluções de boa qualidade segundo seu critério interno, a sua função objetivo. O critério externo tem o objetivo de apenas verificar se as partições são consistentes.

Capítulo 4

Metaheurísticas Propostas

Avaliando os modelos discutidos no capítulo anterior e revisando a literatura sobre os métodos existentes para resolvê-los, observou-se que, para esses problemas, poucas metaheurísticas são propostas. Uma das possíveis razões para isso ocorrer é o alto custo computacional que as metaheurísticas podem apresentar, principalmente para problemas com diversas variáveis, que é o caso das formulações para agrupamento de dados, como as apresentadas no capítulo anterior. Entretanto, vale destacar a qualidade das soluções encontradas por elas, que podem compensar o fato de serem mais lentas e custosas para problemas grandes.

Nesse contexto, em busca de soluções de melhor qualidade optou-se por propor metaheurísticas para esses problemas, já que hoje em dia, a velocidade dos processadores pode compensar a desvantagem do custo computacional das metaheurísticas. Dessa forma, esta Tese apresenta duas metaheurísticas GRASP para os problemas anteriormente discutidos: o de maximização da soma dos pesos das arestas intra-*clusters* e o de maximização da modularidade das partições.

Antes de se entrar em detalhes sobre as características das metaheurísticas propostas, uma introdução sobre a metaheurística GRASP é apresentada.

4.1 Metaheurística GRASP

A metaheurística GRASP (do inglês, *Greedy Randomized Adaptive Search Procedures*) é uma metaheurística de múltiplos inícios, inicialmente proposta por Feo e Resende (1989) para resolver o problema de otimização combinatória conhecido por problema de recobrimento de nós. Essa metaheurística é caracterizada por ser composta por duas fases: a fase construtiva e a fase de busca local. A primeira fase produz uma solução iterativamente até que esta solução seja factível[1]. Essa solução é então melhorada por meio da segunda fase da metaheurística, a de busca local. Cada iteração da metaheurística GRASP pode, portanto, ser sumarizada pelos passos a seguir.

GRASP	
Passo 1.	Construa uma solução pela *Fase Construtiva*.
Passo 2.	Execute a *Busca Local* na solução encontrada no passo anterior e retorne a solução atualizada.
Passo 3.	Se o número máximo de iterações foi atingido pare, senão volte ao Passo 1.

O GRASP foi formalmente definido por Feo e Resende (1995) que apresentaram várias propriedades, assim como sua aplicação a outro problema de otimização combinatória além do de recobrimento de nós, o de designação quadrática.

Desde que foi proposta, diversos métodos para resolver problemas de otimização baseados nessa metaheurística foram documentadas na literatura, como pode ser observado em revisões relacionadas ao tema (Festa e Resende, 2002; Resende e Ribeiro, 2010). Alguns exemplos de problemas para os quais foi proposta pelo menos uma metaheurística GRASP são: agrupamento de dados (Cano et al., 2002; Marinakis et al., 2008), problema de designação quadrática (Oliveira et al., 2004), problema de dimensionamento de lotes (Nascimento et al., 2010a), entre outros. Esses trabalhos são caracterizados pela apresentação de resultados melhores ou bastante competitivos quando comparados a outros métodos encontrados na literatura para os problemas estudados. Tendo a eficiência do GRASP como motivação principal para o seu estudo, optou-se pelo uso dessa metaheurística nesta Tese. Cada uma das suas fases é detalhada a seguir, para uma melhor compreensão das metaheurísticas GRASP propostas.

[1]Uma solução é dita factível se ela obedece todas as restrições do modelo.

4.1.1 Fase Construtiva

A primeira fase do GRASP, a de produção de uma solução, é conhecida como *Fase Construtiva*. Essa fase é caracterizada por um algoritmo semi-guloso[2] que produz uma solução gradualmente, até que ela seja factível segundo a formulação matemática para a qual ela está sendo proposta. Apesar de haver algoritmos na literatura em que a solução produzida pela fase construtiva não é necessariamente factível com o modelo estudado (Nascimento et al., 2010a), a proposta original dessa fase é de que a solução final da fase construtiva seja factível.

A fase construtiva do GRASP é formada por uma lista composta por elementos candidatos a serem incluídos na solução parcial, que é, inicialmente, uma solução vazia. A partir dessa lista, uma outra, conhecida como *Lista Restrita de Candidatos* ou LRC, é construída. A LRC consistirá dos elementos com os melhores custos incrementais. Como ambos os problemas abordados nesta Tese são de maximização, os elementos com melhores custos incrementais serão aqueles que fornecerem os maiores custos incrementais. Tratando essa descrição generalizada da metaheurística para um problema de maximização, a fase construtiva pode ser resumida de acordo com os seguintes passos.

Fase Construtiva	
Passo 1.	Considere inicialmente a solução parcial como uma solução vazia.
Passo 2.	Seja S uma lista contendo todos os candidatos a serem incorporados na atual solução parcial.
Passo 3.	Seja $c(s)$ o custo incremental ao incorporar o elemento $s \in S$ à solução parcial e, c_{min} e c_{max}, respectivamente, os custos incrementais mínimo e máximo dos elementos da lista S.
Passo 4.	Defina a LRC da seguinte maneira: $s \in$ LRC $\Leftrightarrow c(s) \in [c_{min} + \alpha(c_{max} - c_{min}), c_{max}]$, com $\alpha \in [0,1]$.
Passo 5.	Sorteie um elemento da LRC e o incorpore à atual solução parcial.
Passo 6.	Se a atual solução parcial não for factível, retorne para o Passo 2. Senão, vá para o Passo 7.
Passo 7.	Retorne a solução parcial atualizada.

Devido aos Passo 4 e 5, esse procedimento é conhecido por ser semi-guloso. Uma alternativa ao Passo 4, é dada por:

- Ordene a lista S em ordem decrescente e defina a LRC da seguinte maneira: $s \in$ LRC $\Leftrightarrow$ s pertencer à i-ésima posição da lista ordenada S, em que $i < |S|\alpha$, com

[2]De acordo com Hart e Shogan (1987), uma heurística é dita semi-gulosa quando possuir uma componente aleatória, α, em que $\alpha \in [0,1]$. A cada iteração da heurística deve-se decidir entre as $\alpha 100\%$ melhores opções, ou dentre um número c dependente de α de melhores opções.

$\alpha \in [0,1]$.

Finalizada a construção da solução, é iniciada a fase de busca local, que é explicada a seguir.

4.1.2 Busca Local

Uma heurística comumente utilizada para encontrar uma solução para problemas de otimização é conhecida como busca local. Essa heurística é normalmente utilizada como parte de outras metaheurísticas, como GRASP, busca tabu (Glover e Laguna, 1997) e algoritmos meméticos. Dada uma solução x e uma função vizinhança $V : x \rightarrow V(x)$, a busca local procura por uma solução de melhor qualidade nessa vizinhança definida pela função V. Como, nesta Tese, serão apenas abordados modelos de maximização, uma solução de melhor qualidade na $V(x)$ é uma solução cujo valor da função objetivo seja maior do que o valor retornado para a solução x. A busca por uma solução de melhor qualidade é iterativa e a solução x é substituída por uma solução de melhor qualidade da vizinhança da solução da iteração corrente. A busca local pára apenas quando não houver uma solução de melhor qualidade na vizinhança da solução corrente. Os passos básicos de uma busca local estão sumarizados a seguir.

Método da Descida	
Passo 1.	Dada uma solução x, encontre $x' \in V(x)$ tal que $f(x') > f(x)$.
Passo 2.	Se $x' = \emptyset$, vá ao Passo 3, senão, substitua x por x' e volte para o Passo 1.
Passo 3.	Retorne x que é o máximo local.

Essa busca local, conhecida como *método da descida* é uma das abordagens mais usadas de busca local. Outro método bastante conhecido e utilizado é o *método da máxima descida*, que tem uma ligeira modificação nos passos do método anteriormente apresentado e é descrito a seguir.

Método da Máxima Descida	
Passo 1.	Dada uma solução x, encontre $x' \in V(x)$ tal que $f(x') > f(x)$ e $f(x') > f(x'')$, $\forall\ x'' \in V(x)$.
Passo 2.	Se $x' = \emptyset$, vá ao Passo 3, senão, substitua x por x' e volte para o Passo 1.
Passo 3.	Retorne x que é o máximo local.

Pode ser observado que a diferença entre o método da descida e o de máxima descida é que o primeiro procura por qualquer solução x' da vizinhança de x para realizar a troca de soluções, substituindo x por x'. Já o método da máxima descida busca pela *melhor* solução da vizinhança para realizar tal troca de soluções.

Os algoritmos de busca local são conhecidos por terem como limitação o não conhecimento da distância do ótimo global ao local. Apesar de essa característica ser comum à maioria dos métodos heurísticos, o problema da busca local é a grande dependência da solução inicial para gerar soluções de boa qualidade. Por essa razão, tal heurística funciona melhor quando se tem mais de uma solução inicial como ponto de partida para a busca.

No GRASP, um conjunto de soluções é produzido pela sua fase construtiva, que é semi-gulosa, visando fornecer a diversidade necessária das soluções de partida para a busca local. Apesar da diversidade não garantir ótimos locais melhores, há uma maior probabilidade de se obter diferentes ótimos locais quando é fornecido mais de um ponto de partida para a busca. A cada iteração do GRASP, um ótimo local é encontrado, e a melhor solução dentre todas as iterações é mantida como solução final do GRASP.

4.2 GRASP para o Modelo de Maximização da Soma das Arestas Intra-Clusters

Nesta seção é proposta uma metaheurística GRASP para o problema de maximização da soma dos pesos das arestas intra-*cluster*. Esse trabalho deu origem a uma publicação no período *Computers and Operations Research* (Nascimento et al., 2010b). As fases do algoritmo GRASP proposto para essa formulação são apresentadas nas seções seguintes.

4.2.1 Fase Construtiva

A fase construtiva da metaheurística proposta utiliza um procedimento de corte de arestas de um dado grafo G. Inicialmente, a solução é dada pela partição em que todos os nós de G pertencem a um único *cluster*. O processo continua e as arestas do grafo são gradualmente eliminadas de forma que, a cada passo, uma nova componente conexa, que representa um novo *cluster* da partição, seja criada. O procedimento pára quando o número de componentes conexas do grafo atinge o número k de *clusters* desejado, ou seja, quando a solução torna-se factível. A eliminação de arestas do grafo consiste dos seguintes passos.

Fase Construtiva	
Passo 1.	Construa uma lista S, com seus valores em ordem crescente, com as n_h menores arestas dentre todas as componentes conexas C_k do grafo G, em que $n_h = \min\{\max\{0.1m,n\}, \sum_k m_{C_k}\}$. Defina m_{C_k} como o número de arestas da componente conexa C_k.
Passo 2.	Construa a lista LRC usando os $\alpha\|n_h\|$ menores elementos de S.
Passo 3.	Selecione aleatoriamente uma aresta (i,j) e particione a componente conexa C_k, à qual os nós i e j pertencem, em duas componentes conexas, executando o seguinte procedimento: considere $\mathcal{C}_i$ e $\mathcal{C}_j$, respectivamente, os *clusters* aos quais o nó i e j pertencem, inicialmente apenas com estes nós; $\forall t \in C_k$, calcule $\tau = \arg\max_{c=\{i,j\}} w_{tc}$, e elimine todas as arestas $(t,r) \in C_k$, tal que $r \notin C_\tau$.

Inicialmente, C_k é o grafo original G. Note que a lista S tem um número limitado de componentes, aproximadamente 10% do número total de suas arestas. Esse limite foi imposto por dois motivos: melhorar o tempo computacional do algoritmo proposto e garantir uma melhor estabilidade das soluções encontradas por ele. Por meio de testes preliminares, observou-se que esse limite era capaz de encontrar soluções de boa qualidade em um tempo computacional razoável, e, por esse motivo, foi adotado. Vale ressaltar que os passos anteriormente descritos se repetem até que se atinja o número de componentes conexos desejado.

Apesar de a fase construtiva proposta eliminar arestas do grafo, ao passar para fase de busca local, as arestas eliminadas são novamente consideradas para avaliar os movimentos entre os nós dos *clusters*. Se essas arestas não fossem consideradas, não haveria movimento de melhoria da solução encontrada na *Fase Construtiva*.

4.2.2 Busca Local

Depois das soluções terem sido construídas na *Fase Construtiva*, aplica-se esta fase de melhoria. A busca local desenvolvida nesta Tese para ambas as heurísticas GRASP propostas realiza transferências de nós entre *clusters*. Essa estratégia é comumente utilizada e foi inicialmente proposta por Kernighan e Lin (1970). A única diferença entre a heurística de busca local proposta nesta Tese e a de Kernighan e Lin (1970) é que a última permite movimentos que piorem a solução em uma dada iteração.

Para o entendimento da heurística de busca local proposta nesta Tese, considere a Equação 4.1. Essa equação avalia a melhoria da solução encontrada por meio da transferência do nó i de um *cluster* C_t para um *cluster* C_s.

$$\delta_i(C_s) = \sum_{j \in C_s} w_{ij} - \sum_{j \in C_t} w_{ij} \tag{4.1}$$

Os principais passos da busca local proposta são apresentados a seguir. Seja π a partição encontrada na *Fase Construtiva*.

Busca Local	
Passo 1.	Para cada nó $i \in C_t$ da partição π, se $\delta_i(C_k) > 0$, em que $C_k = \arg\max_{C_s \in \pi, C_s \neq C_t} \delta_i(C_s)$, transfira o nó i para o *cluster* C_k, se $C_k \neq \emptyset$.
Passo 2.	Se alguma transferência de nó entre *clusters* foi realizada no Passo 1, então retorne ao Passo 1. Caso contrário, vá para o Passo 3.
Passo 3.	Retorne a partição atualizada π.

Nessa busca local, o Passo 1 é executado no máximo 50 vezes, valor fixado por meio de testes preliminares.

A seguir, a metaheurística GRASP proposta para o modelo de maximização da modularidade de partições é descrita.

4.3 GRASP para o Modelo de Maximização de Modularidade

Esta seção é dedicada à descrição da heurística GRASP proposta para o modelo de maximização da modularidade das partições. Este trabalho foi desenvolvido em colaboração com o Prof Dr Leonidas Pitsoulis. Cada fase da heurística GRASP proposta é detalhada nas seções seguintes.

4.3.1 Fase Construtiva

A fase construtiva do algoritmo proposto tem características opostas às da fase construtiva proposta para o modelo de maximização da similaridade intra-*cluster*. Ao invés do processo, a cada iteração, provocar "quebras" dos *clusters*, os nós são adicionados iterativamente aos *clusters* das partições. Para entender como essa fase funciona, em sua primeira iteração, considere a partição inicial $\pi^{(0)}$ composta por nós isolados de G, i.e., $\pi^{(0)} = \bigcup_{i=1}^{n} C_i$, em que $C_i = \{i\}$. Calcule a modularidade da solução inicial, $q(\pi^{(0)})$, segundo a Equação 3.16. Considere o cálculo da variação da solução dado pela Equação 4.2 ao se combinar um *cluster* i ao *cluster* j, denotado por $\Delta Q(C_i,C_j)$.

$$\Delta Q(C_i,C_j) = \sum_{(i',j'),i'\in C_i,j'\in C_j} r_{i'j'} \tag{4.2}$$

Com isso, a fase construtiva do GRASP proposto para essa formulação matemática tem os seguintes passos.

Fase Construtiva	
Passo 1.	Faça $it \leftarrow 0$. Defina $\pi^{(0)}$ como discutido anteriormente, e faça $\pi'^{(0)} = \emptyset$ e $\pi''^{(0)} = \pi^{(0)}$.
Passo 2.	Para cada *cluster* C_i de $\pi^{(0)}$, insira em uma lista L o valor da maior melhoria resultante de sua combinação com outro *cluster* C_j (custo de combinação), ou seja, o custo de combinação do *cluster* C_i com o $C_j = \arg\max_{C_s\in\pi,C_s\neq C_i} \Delta Q(C_i,C_s)$. Mantenha nessa lista a tripla $(\Delta Q(C_i,C_j), C_i,C_j)$.
Passo 3.	Mantenha a lista L em ordem decrescente e crie um LRC com os $\alpha\|L\|$ maiores custos de combinação, com $\alpha \in [0,1]$.
Passo 4.	Sorteie um elemento $(\Delta Q(C_u,C_v), C_u,C_v)$ da LRC em que: $C_u \in \pi'^{(it-1)}$ e $C_v \in \pi''^{(it-1)}$, se $it > 0$ e $C_u, C_v \in \pi''^{(it-1)}$, caso contrário.
Passo 5.	Se $it = 0$, então faça $\pi''^{(it)} = \{\pi''^{(it-1)} - C_u - C_v\}$. Senão, faça $\pi''^{(it)} = \{\pi''^{(it-1)} - C_v\}$.
Passo 6.	Construa uma nova partição $\pi^{(it)} = \pi'^{(it)} \cup \pi''^{(it)}$ em que $\pi'^{(it)} = \pi'^{(it-1)} \cup \{C_{uv}\}$, e C_{uv} é o *cluster* resultante da combinação dos *clusters* C_u e C_v sorteados no Passo 4.
Passo 7.	Se $\pi'' \neq \emptyset$, faça $it \leftarrow it + 1$ e vá para o Passo 9. Senão, retorne a partição $\pi^{(it)}$.
Passo 8.	Para cada *cluster* isolado de π'', considere a maior melhoria quando ele é adicionado a um *cluster* C_j de π'. A melhoria aqui é denotada por $\delta(C_i,C_j)$, em que C_j pode ser um *cluster* não-vazio de π' ou um *cluster* vazio (nesse último caso, C_i seria simplesmente adicionado à partição π').
Passo 9.	Crie uma lista L com as maiores melhorias para cada *cluster* de π'', sendo que cada elemento da lista é a tripla $(\delta(C_i,C_j),C_i,C_j)$.
Passo 10.	Vá para o Passo 4.

O Passo 9 está explicado de uma maneira ineficiente para facilitar a explicação dessa fase construtiva. Na implementação da heurística, esse passo leva em consideração informação de iterações anteriores, sendo apenas uma atualização da lista L.

4.3.2 Busca Local

A busca local aplicada às soluções encontradas na fase construtiva emprega a mesma estratégia proposta para o GRASP para o problema de maximização da similaridade intra-*cluster*. Entretanto, como a função objetivo é diferente e a escolha do número de *clusters* desse modelo é automática, algumas pequenas modificações foram feitas de forma a se adequar ao modelo de maximização da modularidade das partições. Da mesma forma que a busca local proposta na Seção 4.2.2, ela consiste na transferência de nós entre *clusters* se esta transferência permitir partições com soluções melhores do que a partição corrente. A avaliação da melhoria em se transferir um nó i de um *cluster* C_t para um *cluster* C_s é dada pela Equação 4.3.

$$\delta_i(C_s) = \sum_{j \in C_s} r_{ij} - \sum_{j \in C_t} r_{ij} \tag{4.3}$$

Os passos da busca local são apresentados a seguir.

Busca Local	
Passo 1.	Para cada nó $i \in C_t$ da partição π, se $\delta_i(C_k) > 0$, em que $C_k = \underset{C_s \in \pi, C_s \neq C_t}{\arg\max}\ \delta_i(C_s)$, transfira i para o *cluster* C_k.
Passo 2.	Se algum movimento foi executado no Passo 1, então retorne ao Passo 1. Senão, vá ao Passo 3.
Passo 3.	Retorne a partição atualizada π.

Assim como na busca local proposta para o modelo de maximização de pesos das arestas intra-*cluster*, o Passo 1 é aqui executado no máximo 50 vezes.

As principais diferenças entre essa busca local e a busca local para o modelo anteriormente apresentado é que C_k pode ser um *cluster* vazio e que o número de *clusters* inicialmente definido na fase construtiva pode aumentar ou diminuir com as iterações da busca local.

4.4 Considerações Finais

Este capítulo apresentou as metaheurísticas GRASP propostas para os modelos de maximização da similaridade intra-*cluster* e de maximização da modularidade das partições. Os motivos pelos quais optou-se por essa metaheurística foram a experiência que a aluna tem com essa metaheurística, e o fato dela ser uma metaheurística que tem apresentado bons resultados para diversos tipos de problemas. É observado na literatura que o GRASP,

quando hibridizado, apresenta um melhor desempenho. Normalmente a hibridização é feita com a metaheurística *path-relinking* (Crowston et al., 1963; Glover e Laguna, 2000). Por esse motivo, foram testadas hibridizações das metaheurísticas GRASP propostas com *path-relinking*. Entretanto, a melhoria das soluções encontradas por essas hibridizações com relação à metaheurística GRASP pura não foi relevante, quase inexistente, para os problemas aqui apresentados. A deficiência pode estar associada ao *path-relinking* proposto, mas como não foi encontrada nenhuma alternativa que melhorasse a sua qualidade, a hibridização do GRASP não foi e nem será discutido nesta Tese.

A seguir, são propostos novos modelos matemáticos para agrupamento de dados em grafos. Eles consistem da modificação de diferentes versões de uma medida conhecida por avaliar a conectividade de nós em grafos, o *clustering coefficient*. Adicionalmente, um algoritmo multi-nível é proposto para resolver esses modelos de forma heurística.

Capítulo 5

Proposta de *Clustering Coefficient*

A relevância do problema de agrupamento de dados é destacada pela grande quantidade de estudos e algoritmos para a sua solução. Isso acontece pois, apesar da diversidade de algoritmos para solucioná-lo, cada um deles possui diferentes limitações. A principal causa disso é a dificuldade em se definir uma formulação matemática que generalize esse problema. Por essa razão, o problema ainda necessita de novas alternativas e formulações que o resolvam ou que encontrem partições por meio de algoritmos que possuam uma base matemática mais sólida que minimizem as limitações dos já existentes.

Uma das limitações encontradas em algoritmos de agrupamento é a necessidade da definição prévia do número de *clusters* na partição a ser encontrada. Nesta Tese, os algoritmos capazes de definir a partição e o número de *clusters* de um grafo automaticamente são chamados de algoritmos de agrupamento automáticos em grafos. Esses algoritmos são de grande importância para a análise de um conjunto de dados, pois permitem a aplicação de algoritmos de agrupamento a um conjunto de dados sem um conhecimento prévio de sua configuração, que é justamente a principal característica dos dados utilizados no problema de agrupamento, um problema da área de aprendizado não-supervisionado. Por essa razão, a realização de pesquisas envolvendo esse tipo de algoritmo é de clara importância para o problema de agrupamento de dados.

Algoritmos de agrupamento automáticos que sejam baseados em um modelo matemático bem definido não são muito explorados nas pesquisas da área de aprendizado não-supervisionado. Usualmente, nesses algoritmos, adota-se um estudo empírico do conjunto de dados, como se o problema de detectar o número de *clusters* presentes no con-

junto de dados fosse separado do problema de encontrar a partição ideal. Exemplos dessa tendência podem ser encontrados em estudos relacionados à teoria espectral, em que os autores definem o número de *clusters* por meio da escolha do índice k do autovalor que produza o maior *eigengap*[1] da matriz Laplaciana. Em seguida é aplicado algum algoritmo de agrupamento com o número de *clusters* previamente definido pelo *eigengap*.

Um exemplo de modelo de agrupamento automático em grafo que não trate o problema de definir o número de *clusters* da partição a ser encontrada separadamente do de encontrar a partição final é o modelo de maximização de modularidade das partições (Girvan e Newman, 2002). O que torna o processo automático nesse modelo é a matriz de modularidade, R, possuir valores negativos, quando conveniente, que seriam as "penalidades" de dois nós pertencerem a um mesmo *cluster*. Além disso, o mesmo modelo tem uma vantagem não encontrada em muitos algoritmos e modelos de agrupamento de dados: a possibilidade da avaliação da qualidade das partições geradas, sempre tendo um limitante superior (ótimo) para ser atingido, o que não é uma questão simples e direta para modelos de agrupamento.

Além de propor metaheurísticas para modelos já existentes de agrupamento em grafos, a fim de analisar as partições encontradas por eles, o principal objetivo desta Tese é propor um novo modelo matemático que trate das duas limitações encontradas nos algoritmos de agrupamento de dados em grafos discutidas anteriormente, ou seja, que defina automaticamente o número de *clusters* e forneça a qualidade da partição encontrada segundo o critério adotado pelo algoritmo. Se o uso de um algoritmo ótimo for inviável para verificar a validade das soluções, uma heurística para tal modelo deve ser proposta para avaliar sua capacidade de particionar grafos de forma coerente com um dado critério.

Para cumprir essa meta, como ponto de partida para propor uma nova medida de avaliação de partições, estudou-se uma medida conhecida por avaliar a tendência de agrupamento de nós de um grafo, conhecida como *clustering coefficient* (Watts e Strogatz, 1998). Existem duas versões dessa medida, uma global e outra local. A primeira avalia a tendência de conectividade do *grafo*, enquanto a segunda foi desenvolvida para analisar a tendência de conectividade dos *nós* do grafo individualmente, por meio da análise da conectividade de sua vizinhança. Nesta Tese, a segunda versão do *clustering coefficient* é o ponto de partida do modelo proposto para agrupamento de dados em grafos. A seguir, uma introdução à medida *clustering coefficient* é apresentada.

[1] $k = \arg\max_{1 \leq i \leq n-1} eigengap_i$, em que $eigengap_i = |\lambda_i - \lambda_{i+1}|$.

5.1 Clustering Coefficient

Antes de abordar a medida *clustering coefficient*, vale informar que, neste capítulo, o grau de um dado nó de um grafo é definido como o número de nós adjacentes a ele, independente do grafo ter peso ou não em suas arestas. Dessa forma, a definição de grau deste capítulo é a dada pela Equação 2.1.

Como já foi dito, o *clustering coefficient* é uma medida que pode ser encontrada na literatura em duas variações: a global e a local. O *clustering coefficient* global analisa, em um dado grafo, a razão entre o seu número de ciclos de três nós, denominados triângulos, e o seu número de triplas. Uma tripla é um subgrafo conexo de três nós, ou seja, três nós do grafo que são conectados por duas ou três arestas (Opsahl, 2009). O *clustering coefficient* global pode ser medido pela Equação 5.1.

$$\mathcal{C}(G) = \frac{\text{número de triângulos de } G}{\text{número de triplas de } G} \tag{5.1}$$

É importante mencionar que essa definição é válida apenas para grafos não dirigidos e sem pesos. Algumas propostas para a sua generalização para grafos dirigidos e ponderados são discutidas em (Opsahl, 2009). Entretanto, como o *clustering coefficient* global não foi estudado para propor o modelo que está sendo proposto nesta Tese, esse tema não será mais abordado daqui em diante.

O segundo tipo de *clustering coefficient*, o local, foi proposto por Watts e Strogatz (1998). Como essa será a abordagem adotada nesta Tese, quando for mencionado *clustering coefficient*, deve ser assumido o tipo local. A sua definição surgiu do estudo da conectividade dos vizinhos de um dado nó. Esse estudo avalia, para cada um dos vizinhos de um dado nó i, a fração entre o número de conexões existentes entre eles e o número total de possíveis conexões entre os vizinhos do nó, este último sendo igual ao número de arestas em um clique de g_i nós: $g_i(g_i - 1)/2$. A seguir, uma equação simplificada para calcular o *clustering coefficient* de um dado nó i de um grafo sem pesos G é apresentada.

$$\mathcal{C}_i(G) = \frac{\text{número de arestas compartilhadas entre os vizinhos do nó } i}{\text{número total de possíveis conexões entre os vizinhos do nó } i} \tag{5.2}$$

O denominador da Equação 5.2, que se refere ao número total de possíveis conexões entre os vizinhos do nó i, leva em consideração apenas a informação local do grau do nó i para estimar o possível clique entre os vizinhos. Uma equação que faz uso da matriz de adjacências para calcular o *clustering coefficient* de um nó i, $\mathcal{C}_i(G)$, é dada por:

$$\mathcal{C}_i(G) = \frac{2\sum_{\substack{j=1\\ j\neq i}}^{n-1}\sum_{\substack{k=j+1\\ k\neq i}}^{n} a_{ij}a_{jk}a_{ik}}{g_i(g_i-1)}. \tag{5.3}$$

O *clustering coefficient* foi inicialmente proposto apenas para grafos sem pesos. Anos depois, sugiram algumas generalizações dessa medida para grafos com pesos. A primeira delas, proposta por Barrat et al. (2004), é apresentada na Equação 5.4.

$$\mathcal{C'}_i(G) = \frac{1}{d_i(g_i-1)}\sum_{\substack{j=1\\ j\neq i}}^{n-1}\sum_{\substack{k=j+1\\ k\neq i}}^{n}\frac{(w_{ij}+w_{ik})}{2}a_{ij}a_{jk}a_{ik}, \tag{5.4}$$

Segundo Onnela et al. (2005), um problema associado à Equação 5.4 é ser indiferente ao peso das arestas (j,k) que estão conectando os vizinhos do nó i. Isso significa que, se os pesos das arestas entre os vizinhos do nó i forem bem inferiores ao peso de suas arestas conectando ao nó i, o seu *clustering coefficient* não irá diferir do caso em que os pesos das arestas entre os vizinhos do nó i forem bem altos. Como alternativa a essa medida, no ano seguinte, Onnela et al. (2005) apresentaram sua generalização para grafos ponderados, considerando o peso das arestas entre os vizinhos do nó i. Nessa medida, é necessário modificar o numerador da formulação original para grafos sem pesos, que indica o número de triângulos em torno do nó i, pela soma das intensidades dos triângulos ao redor do nó i. Tal formulação, $\tilde{\mathcal{C}}_i(G)$, leva em consideração o peso das arestas entre os vizinhos do nó i e é dada por:

$$\tilde{\mathcal{C}}_i(G) = \frac{2\sum_{\substack{j=1\\ j\neq i}}^{n-1}\sum_{\substack{k=j+1\\ k\neq i}}^{n}(\tilde{w}_{ij}\tilde{w}_{ik}\tilde{w}_{jk})^{1/3}}{g_i(g_i-1)}, \tag{5.5}$$

em que $\tilde{w}_{ij} = w_{ij}/\max\{w_{ij}|i,j \in \mathbb{Z}, 1 \leq i,j \leq n\}$. Pode ser observado que $\tilde{\mathcal{C}}_i(G) \in [0,1]$ já que $\sum_{\substack{j=1\\ j\neq i}}^{n-1}\sum_{\substack{k=j+1\\ k\neq i}}^{n}(\tilde{w}_{ij}\tilde{w}_{ik}\tilde{w}_{jk})^{1/3} \leq g_i(g_i-1)/2$. Se $\tilde{\mathcal{C}}_i(G) = 1$, então a vizinhança do nó i inclui todas as combinações de triângulos que podem ser formadas, em que cada aresta que faz parte dos triângulos tem peso máximo.

Convencionalmente, quando se deseja calcular o valor do *clustering coefficient* de um grafo, calcula-se a média do *clustering coefficient* entre todos os seus nós. Portanto, o *clustering coefficient* de um grafo sem pesos pode ser calculado como a seguir:

$$\mathcal{C}(G) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{C}_i(G). \tag{5.6}$$

Analogamente, o *clustering coefficient* de um grafo com pesos possui duas versões,

cada uma devido às diferentes generalizações existentes para grafos com pesos. A primeira, proposta por Barrat et al. (2004), é apresentada a seguir:

$$\mathcal{C}'(G) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{C}'_i(G). \tag{5.7}$$

O *clustering coefficient* de um grafo, segundo a versão proposta por Onnela et al. (2005), pode ser calculada como a seguir:

$$\tilde{\mathcal{C}}(G) = \frac{1}{n}\sum_{i=1}^{n}\tilde{\mathcal{C}}_i(G). \tag{5.8}$$

Para se avaliar partições de um grafo, no capítulo seguinte é proposta uma adaptação do *clustering coefficient* das Equações 5.3, 5.4 e 5.5. Com as medidas propostas, busca-se detectar as partições que tenham um valor elevado para essas medidas. Para isso, são introduzidas formulações matemáticas que procuram maximizar as medidas propostas. A seguir, essas medidas de avaliação de partições são apresentadas.

5.2 Clustering Coefficient para Avaliar Partições de um Grafo

Como já foi mencionado, esta seção propõe medidas baseadas nas diferentes variações de *clustering coefficient* para avaliar a tendência de agrupamento de uma partição. Além disso, são propostas formulações matemáticas para encontrar partições que maximizem essas medidas propostas.

Para facilitar a compreensão das medidas propostas nesta Tese, considere uma partição do grafo. Nessa partição, elimine as arestas que conectam os diferentes *clusters*, ou seja, faça o corte do grafo de acordo com a partição dada. Segundo a medida de avaliação proposta, o valor do *clustering coefficient* dessa partição é o valor do *clustering coefficient* calculado pela Equação 5.6, se o grafo não é ponderado, e pelas Equações 5.7 e 5.8, caso o grafo seja ponderado. Vale observar que ambas as medidas avaliam a conectividade dos vizinhos dos nós de um grafo para verificar se eles têm uma tendência de agrupamento.

Tendo em vista o que foi dito, para se calcular o *clustering coefficient* de um nó de uma partição π em um grafo sem pesos em suas arestas, considere a Equação 5.9.

$$\mathcal{H}_i(\pi) = \frac{2\sum_{\substack{j=1\\ j\neq i}}^{n-1}\sum_{\substack{k=j+1\\ k\neq i}}^{n} a_{ij}a_{jk}a_{ik}}{g'_i(g'_i-1)}y_{ijk} \tag{5.9}$$

em que y_{ijk} é uma variável binária que assume valor 1 se os nós i, j e k pertencem ao mesmo *cluster* e 0, caso contrário; e g'_i pode ser calculado pela Equação 5.10.

$$g'_i = \sum_{j=1}^{n} a_{ij} x_{ij}, \tag{5.10}$$

em que x_{ij} é uma variável binária que recebe valor 1 se os nós i e j pertencem ao mesmo *cluster* e 0, caso contrário. Pode ser observado que g'_i é o número de nós adjacentes ao nó i que estão no mesmo *cluster*. Note que *clusters* com nós isolados ou com apenas uma conexão não podem ser analisados por meio dessa medida, pois o denominador seria $g'_i(g'_i - 1) = 0$, resultando em uma divisão por zero. Dessa forma, quando isso ocorrer, considere $g'_i = 2$, pois a medida fará sentido e indicará que o nó não tem nenhuma tendência de agrupamento.

Para avaliar a qualidade de uma partição π de um grafo com pesos, duas equações podem ser utilizadas. A primeira, baseada na formulação proposta por Barrat et al. (2004), considera o *clustering coefficient* de um nó i de π dado por:

$$\mathcal{H}'_i(\pi) = \frac{1}{d_i(g'_i - 1)} \sum_{\substack{j=1 \\ j \neq i}}^{n-1} \sum_{\substack{k=j+1 \\ k \neq i}}^{n} \frac{(w_{ij} + w_{ik})}{2} a_{ij} a_{jk} a_{ik} y_{ijk}. \tag{5.11}$$

A segunda calcula o *clustering coefficient* de uma partição π de um grafo com pesos considerando a medida proposta por Onnela et al. (2005). Nesse caso, o *clustering coefficient* de um nó i da partição π é dado por:

$$\tilde{\mathcal{H}}_i(\pi) = \frac{2 \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{\substack{k=j+1 \\ k \neq i}}^{n} (\tilde{w}_{ij} \tilde{w}_{jk} \tilde{w}_{ik})^{1/3}}{g'_i(g'_i - 1)} y_{ijk}. \tag{5.12}$$

Analogamente ao cálculo do *clustering coefficient* de um grafo, o *clustering coefficient* de uma partição é dado pela sua média para todos os nós da partição. Portanto, o *clustering coefficient* de uma partição em um grafo sem pesos é dado por: $\mathcal{H}(\pi) = \sum_{i=1}^{n} \mathcal{H}_i(\pi)/n$, enquanto o *clustering coefficient* de uma partição em um grafo com pesos é dado por: $\mathcal{H}'(\pi) = \sum_{i=1}^{n} \mathcal{H}'_i(\pi)/n$ ou $\tilde{\mathcal{H}}(\pi) = \sum_{i=1}^{n} \tilde{\mathcal{H}}_i(\pi)/n$. Vale observar que o valor dessas medidas pertence ao intervalo [0,1].

A seguir, são apresentadas formulações não-lineares para encontrar partições que maximizem as medidas propostas.

5.3 Formulação Matemática

O objetivo deste capítulo é propor uma formulação matemática que encontre uma partição em um grafo utilizando a conectividade dos nós do grafo a ser particionado. Essa formulação deve ser capaz de identificar a partição do grafo sem a necessidade de definir o número de grupos. Baseado na medida previamente proposta, um modelo para a maximização do *clustering coefficient* das partições é proposto a seguir:

$$\max \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \frac{h_{ijk}}{g'_i(g'_i-1)} y_{ijk}$$

sujeito a:

$$x_{ij} + x_{jk} \geq 2y_{ijk} \qquad 1 \leq i,j \leq n, j+1 \leq k \leq n \tag{5.13}$$

$$x_{ij} + x_{jk} - 2x_{ik} \leq 1 \qquad 1 \leq i,j,k \leq n \tag{5.14}$$

$$g'_i = \sum_{j=1}^{n} x_{ij} a_{ij} \qquad 1 \leq i \leq n \tag{5.15}$$

$$x_{ij}, y_{ijk} \in \{0,1\} \qquad 1 \leq i,j \leq n, j+1 \leq k \leq n \tag{5.16}$$

$$\tag{5.17}$$

em que $h_{ijk} = a_{ij}a_{ik}a_{jk}$ se o grafo não tiver pesos e $h_{ijk} = (\tilde{w}_{ij}\tilde{w}_{jk}\tilde{w}_{ik})^{1/3}$ se o grafo tiver pesos em suas arestas; y_{ijk} é uma variável binária que assume valor 1 se os nós i, j e k pertencem ao mesmo *cluster* e 0, caso contrário; e x_{ij} é uma variável binária que assume valor 1, se os nós i e j pertencem ao mesmo *cluster* e 0, caso contrário.

As Restrições (5.13) fazem com que se x_{ij} ou x_{jk} for(em) nula(s), a variável y_{ijk} também seja nula. Além disso, essas restrições estabelecem que se as variáveis x_{ij} e x_{jk} forem ambas unitárias, a variável y_{ijk} também será unitária, pois, nesse caso, $y_{ijk} \leq 1$ e o problema se torna de maximização de valores positivos. As Restrições (5.14) fazem com que a variável x_{ik} seja unitária se e somente se as variáveis x_{ij} e x_{jk} também forem unitárias. As Restrições (5.15) obrigam as variáveis g'_i a assumir o valor equivalente ao número de nós adjacentes ao nó i que compartilham o mesmo *cluster*. Por fim, as Restrições (5.16) fazem com que as variáveis x_{ij} e y_{ijk} sejam binárias.

Outra formulação, baseada no índice *clustering coefficient* para grafos com pesos proposto por Barrat et al. (2004), é apresentada a seguir. Esse modelo também é não-linear e suas variáveis são inteiras.

$$\max \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \frac{h_{ijk}}{d_i(g'_i-1)} y_{ijk}$$

sujeito a:

$$x_{ij} + x_{jk} \geq 2y_{ijk} \qquad 1 \leq i,j \leq n, j+1 \leq k \leq n \tag{5.18}$$

$$x_{ij} + x_{jk} - 2x_{ik} \geq 1 \qquad 1 \leq i,j,k \leq n \tag{5.19}$$

$$g'_i = \sum_{j=1}^{n} x_{ij} a_{ij} \qquad 1 \leq i \leq n \tag{5.20}$$

$$x_{ij}, y_{ijk} \in \{0,1\} \qquad 1 \leq i,j \leq n, j+1 \leq k \leq n \tag{5.21}$$

em que $h_{ijk} = \frac{(w_{ij}+w_{ik})}{2} a_{ij} a_{jk} a_{ik}$. Note que as restrições desse modelo são as mesmas do modelo anterior.

Vale a pena ressaltar que muitas variáveis e restrições de ambos os modelos matemáticos apresentados nesta seção podem ser eliminadas. Para tal, considere $i < j$ para x_{ij} e $i < j < k$ para y_{ijk} e faça alguns ajustes nas restrições e função objetivo. A formulação foi apresentada dessa maneira para torná-la mais intuitiva ao leitor.

Como essa formulação é não-linear e, pelo conhecimento da aluna, ele não se enquadra em nenhum caso especial de problemas não-lineares que podem ser resolvidos de maneira eficiente, apenas uma busca exaustiva poderia encontrar sua solução ótima. Entretanto, como essa busca é intratável, pois o espaço de solução pode ser grande, foi proposta uma heurística multi-nível para resolvê-lo, que é explicada em detalhes na próxima seção.

5.4 Heurística Multi-Nível

Para resolver heuristicamente o modelo proposto na seção anterior, diferentes abordagens foram consideradas. Inicialmente, uma metaheurística GRASP semelhante à proposta para a maximização da modularidade foi testada. Foi observado que, para problemas com mais de 500 nós, essa metaheurística não se mostrou uma boa opção, pois o seu custo computacional é muito alto devido à complexidade do algoritmo utilizado para calcular uma solução factível, que é $O(n^3)$. Dessa forma, passou-se a procurar alternativas eficientes para resolver esse problema. As opções adequadas para a resolução do problema considerando sua complexidade foram: a adoção de heurísticas paralelas e/ou daquela classe de algoritmos discutida na Seção 3.2, de algoritmos multi-níveis. Por se sentir mais

confortável com algoritmos multi-níveis, a aluna optou pela segunda opção.

Como já foi observado, os algoritmos multi-níveis têm sido amplamente usados para encontrar partições de boa qualidade (Karypis e Kumar, 1996, 1998; Dhillon et al., 2007; Noack e Rotta, 2009; Oliveira e Seok, 2007, 2008). Eles são caracterizados por apresentar três fases: a de contração dos nós, a de particionamento do grafo contraído e a de refinamento. Na última ocorre a expansão do grafo e o simultâneo melhoramento da partição encontrada na fase anterior. Nesta Tese, um algoritmo multi-nível é proposto para resolver o modelo introduzido na seção anterior. As fases desse algoritmo são compostas pelos passos apresentados a seguir:

1. *Fase de Contração:* Nessa fase, uma seqüência de emparelhamentos maximais entre os nós do grafo são executados até o grafo atingir um tamanho pré-definido ou não ser mais possível emparelhar seus nós.

2. *Fase de Particionamento:* Nessa fase, o grafo contraído na fase anterior é particionado por algum algoritmo de particionamento de grafos. Para isso, foram testadas diversas alternativas, inclusive as metaheurísticas GRASP apresentadas nesta Tese e alguns algoritmos espectrais (Von Luxburg, 2007). Entretanto, a melhor opção foi o algoritmo METIS (Karypis e Kumar, 1996, 1998), que apresentou partições com *clusters* melhor balanceados e que foram mais facilmente refinados na fase posterior.

3. *Fase de Refinamento:* Por fim, a partição encontrada na fase anterior é melhorada pela simultânea aplicação da busca local e expansão dos nós do seu grafo. Esse processo continua até o grafo particionado ser o grafo original e a partição ser o melhor local.

   Os detalhes de cada uma das fases são apresentados a seguir.

5.4.1 Fase de Contração

A fase de contração do algoritmo multi-nível proposto nesta Tese aplica emparelhamentos maximais aos nós do grafo a ser particionado. Antes de discutir os algoritmos envolvidos nesse processo, o problema de emparelhamento e a justificativa para o seu uso em algoritmos multi-níveis são apresentados.

Um emparelhamento em um grafo G é definido por um conjunto S de arestas em que, para cada $(i,j) \in S$, não há nenhuma aresta (k,j), com $k \neq i$ ou (k,i) com $k \neq j$ que pertença a S. Um exemplo de emparelhamento é ilustrado na Figura 5.1.Na Figura 5.1, as arestas em destaque são as que compõem o conjunto S. Note que esse conjunto corresponde a um emparelhamento maximal, pois para quaisquer de suas arestas, uma de suas

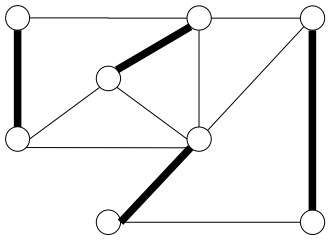


Figura 5.1: Grafo com seu emparelhamento, representado pelo conjunto de arestas destacadas por linhas mais espessas.

pontas está contida em alguma aresta pertencente ao conjunto S. Um emparelhamento é dito máximo se o emparelhamento for maximal e tiver o maior número de arestas possível dentre todos os emparelhamentos maximais. Note que essa definição é relativa a grafos sem pesos em suas arestas. Em um grafo com pesos, basta substituir, nessa definição, o número de arestas pelo peso de arestas. Para encontrar a solução do problema de emparelhamento máximo em um grafo com pesos, existem algoritmos polinomiais da ordem $O(n^4)$ (Papadimitriou e Steiglitz, 1998) e $O(n^3)$ (Gabow, 1973). Como se pretende executar o emparelhamento maximal mais de uma vez, se o grafo tiver um número de nós elevado, esses algoritmos que encontram o emparelhamento maximal máximo, mesmo sendo polinomiais, seriam inviáveis. Dessa forma, assim como tem sido feito na literatura (Barnard e Simon, 1994; Karypis e Kumar, 1998), foram procurados diferentes emparelhamentos maximais por meio de algum dos algoritmos existentes para tal.

É importante discutir a decisão de se contrair o grafo por meio de um algoritmo de emparelhamento. A justificativa para o seu uso, que pode ser encontrada em (Karypis e Kumar, 1998), é que o grafo contraído por emparelhamento preserva muitas das propriedades do grafo original (um exemplo é a planaridade do grafo[2]).

Para realizar os emparelhamentos maximais, foi usado um procedimento baseado em uma abordagem aleatória. Nessa abordagem, um nó i que ainda não foi emparelhado é escolhido. Então, o nó j adjacente a i que ainda não foi emparelhado em iteração anterior e que possui o maior peso na aresta que o conecte com o nó i (dentre os possíveis candidatos a emparelhamento com o nó i) é escolhido. Se o nó j existir, i e j são emparelhados. Caso o grafo não tenha peso em suas arestas, esse procedimento seleciona aleatoriamente um nó e escolhe um de seus nós adjacentes, que ainda não foi emparelhado, para ser emparelhado nesta iteração. Uma vantagem da abordagem aleatória é a produção de um conjunto diverso de soluções.

[2]Um grafo é dito planar se ele puder ser desenhado em um plano de forma que nenhuma de suas arestas interceptem outra (Papadimitriou e Steiglitz, 1998).

A cada emparelhamento, os pesos das arestas entre cada par de nós e seus nós adjacentes são atualizados, mesmo em grafos sem peso. Por exemplo, considere um par de nós i e j que será emparelhado. Os pesos das arestas entre o nó i e seus nós adjacentes terão agora que levar em consideração os pesos das arestas entre esses nós adjacentes e o outro nó do emparelhamento, o nó j. A forma mais comum de atualizar os pesos de grafos emparelhados para algoritmos multi-níveis ocorre da seguinte maneira: dada uma aresta adjacente a i (ou j), denominada k, soma-se o peso das arestas conectando o nó i (ou j) com k e j (ou i) com k. Se o grafo não tiver peso em suas arestas, considere o mesmo procedimento atribuindo o valor 1 aos pesos das arestas existentes entre o par de nós, e 0, caso contrário.

Nesta Tese, a fase de contração é executada até que o número de nós do grafo inicial, que será inicialmente particionado na fase de particionamento, seja inferior a $\max\{\min\{100,n\},0.2n\}$. Isso significa que, se o grafo já tiver um número de nós inferior ou igual a 100, apenas uma iteração do emparelhamento é executada. Esse grafo contraído, conhecido como *grafo base* ou *grafo inicial*, é particionado na fase seguinte do algoritmo multi-nível, a fase de particionamento. Tal fase é detalhada a seguir.

5.4.2 Fase de Particionamento

Para particionar o grafo base, diversas abordagens e algoritmos podem ser adotados. Foram testadas diversas delas, incluindo as heurísticas GRASP anteriormente propostas, alguns algoritmos espectrais e o algoritmo METIS detalhado na Seção 3.2. Depois de testes preliminares com essas opções, decidiu-se pelo algoritmo METIS, por ser rápido e prover melhores partições que outros algoritmos de particionamento testados. Por isso, os resultados usando outros algoritmos não são reportados.

Como o algoritmo METIS exige a definição prévia do número de *clusters*, decidiu-se por gerar um conjunto de partições do grafo base com diferentes números de *clusters*. Tal conjunto é composto por partições com $k = \{2,\ldots,n/3\}$ *clusters*. Como a fase de contração apresenta características aleatórias, os emparelhamentos maximais gerados a cada iteração podem ser diferentes. Para estabilizar os resultados, foram executados 5 emparelhamentos maximais. Dessa forma, para cada k-partição do grafo base, foram geradas 5 partições, na tentativa de se garantir uma solução final de melhor qualidade. Todos esses números foram obtidos por meio de testes preliminares, levando-se em consideração o tempo computacional e a qualidade da partição final.

5.4.3 Fase de Refinamento

Para essa fase, foi proposta uma busca local, que é aplicada a toda partição encontrada na fase de particionamento. Diferentes procedimentos de busca local foram testados para o algoritmo proposto, e, novamente, foi escolhido aquele que apresentou a melhor combinação de eficiência e desempenho.

Esse algoritmo foi elaborado para resolver heuristicamente os modelos propostos neste capítulo, que apresentam três variações: uma para grafos sem pesos e duas para grafos com pesos. Dessa forma, a única diferença dessa fase para as três formulações é o cálculo de um valor de transferência, aqui denominado t_i. Esse cálculo considera as seguintes possibilidades:

- Se o grafo a ser particionado, G, não tiver pesos em suas arestas, considere $t_i = \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \frac{a_{ij}a_{ik}a_{jk}}{g'_i(g'_i-1)} y_{ijk}$.
- Se G for um grafo com pesos em suas arestas e deseja-se resolver o modelo tendo como ponto de partida o *clustering coefficient* proposto por Barrat et al. (2004), considere $t_i = \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \frac{(w_{ij}+w_{ik})a_{ij}a_{jk}a_{ik}}{2d_i(g'_i-1)} y_{ijk}$.
- Se G for um grafo ponderado e deseja-se resolver o modelos proposto tendo como ponto de partida a medida proposta por Onnela et al. (2005), considere $t_i = \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \frac{(\tilde{w}_{ij}\tilde{w}_{ik}\tilde{w}_{jk})^{1/3}}{g'_i(g'_i-1)} y_{ijk}$.

A fase de refinamento proposta pode ser resumida nos passos apresentados a seguir.

Fase de Refinamento	
Passo 1	Leia o grafo e a partição que será refinada. Faça $it \leftarrow 0$.
Passo 2	Desmarque todos os nós do grafo.
Passo 3	Escolha o nó não-marcado i cujo t_i seja menor do que t'_i. t'_i é calculado da mesma maneira de t_i, modificando-se as variáveis como se o nó i pertencesse a um *cluster* c, diferente do *cluster* ao qual ele pertence. Marque o nó i. Se $n < 500$, vá ao Passo 4, senão, vá ao Passo 5.

Continua na próxima página...

Passo 4	Se transferir o nó i para o *cluster* c melhora a solução atual, vá para o Passo 5. Senão, vá para o Passo 6.
Passo 5	Mova o nó i para o *cluster* c.
Passo 6	Se todos os nós estiverem marcados, faça $it \leftarrow it + 1$ e vá para o Passo 7, senão, vá para o Passo 5.
Passo 7	Se não houver nenhuma melhoria nesta iteração ou $it >$MAX-ITERACOES, então vá para o Passo 8. Senão, vá ao Passo 2.
Passo 8	Retorne a solução mais atualizada.

Note que, segundo os passos do procedimento de busca local anteriormente apresentados, quando o grafo tiver mais de 500 nós, o Passo 3 é eliminado. Isso acontece, pois esse passo de verificação da melhoria da solução pode ser bem custoso. Entretanto, para grafos menos que 500 nós, foi observado que é viável considerar esse passo de verificação antes de mover o nó i de *cluster*. Vale mencionar que essa busca local permite que o número de *clusters* da partição inicial diminua, pois a transferência de nós entre *clusters* pode esvaziar algum desses *clusters*. O valor adotado para o pârametro MAX-ITERACOES foi 10, escolhido por meio de testes preliminares.

Tendo sido descritas as três frases do algoritmo multi-nível, os seus passos podem ser resumidos da seguinte forma, considerando como entrada G, o grafo a ser particionado. Esse algoritmo é referido como CCML.

CCML	
Passo 1	(*Fase de Contração*) Contraia o grafo G assim como foi apresentado na Seção 5.4.1. O resultado é um conjunto de 5 grafos base.
Passo 2	(*Fase de Particionamento do grafo base*) Para cada grafo base, encontre o conjunto de partições para cada k, com $k = \{2, \ldots, n/3\}$, pelo algoritmo METIS, como explicado na Seção 5.4.2.
Passo 3	(*Fase de refinamento*) Aplique a busca local apresentada na Seção 5.4.3 a todas as partições encontradas na fase anterior, a de particionamento. Retorne a partição que produzir a melhor solução final.

A seguir, no capítulo de experimentos computacionais, a estabilidade, o desempenho e uma análise individual e comparativa considerando todos os algoritmos propostos nesta Tese são apresentados.

Capítulo 6

Testes Computacionais

Para avaliar o desempenho dos algoritmos propostos nesta Tese, uma série de experimentos foi realizada com grafos artificiais e reais utilizando diversas configurações. Inicialmente, como todas as metaheurísticas aqui propostas têm uma componente aleatória em seus procedimentos, uma análise da robustez desses algoritmos foi realizada. Em seguida, para verificar a eficiência das metaheurísticas, elas foram comparadas com outros algoritmos encontrados na literatura. Por fim, as metaheurísticas propostas para diferentes formulações foram comparadas entre si a fim de avaliar para quais tipos de grafos cada uma delas é mais indicada, ou seja, gera partições de melhor qualidade.

Para melhor diferenciar os métodos propostos, o GRASP proposto para o modelo de maximização da similaridade intra-*cluster* é chamado de GRASPI. Já o GRASP proposto para o modelo de maximização da modularidade das partições é denominado GRASPII. O algoritmo multi-nível proposto para a maximização da medida baseada no *clustering coefficient* de Onnela et al. (2005) será denotado por CCMLI, enquanto que a versão baseada em Barrat et al. (2004) será denominada CCMLII.

Em todos os testes realizados, os parâmetros das metaheurísticas GRASP foram padronizados. Considerou-se um total de 100 iterações e o parâmetro α das suas fases construtivas foi gerado aleatoriamente entre [0,1]. Para o método multi-nível, foram utilizados os parâmetros discutidos no Capitulo 5.

A seguir, o grafos modulares[1] gerados para os testes são discutidos, descrevendo

[1]Grafos modulares são grafos com tendência de agrupamento em sua estrutura.

suas configurações e método de geração.

6.1 Configuração dos Dados

Grafos modulares aleatoriamente gerados com diversas configurações foram testados para avaliar a qualidade dos métodos propostos. Para gerar tais grafos, foi usada a função *SimDataAffiliation* da biblioteca *Statistical Inference for Modular Networks* (SIMoNe), em versão para R-*project*. Essa função produz um grafo modular usando uma simulação de uma amostra Gaussiana ao qual o grafo é associado. Os parâmetros de tal função são: o número de nós do grafo a ser gerado (n), o número de amostras Gaussianas para a simulação (a), a probabilidade de existir uma aresta entre dois nós intra-*cluster* ($p1$), a probabilidade de haver uma aresta entre um par de nós de *clusters* diferentes ($p2$), um vetor que fornece a proporção do número de objetos por *cluster* (c) e a probabilidade de haver nós isolados (i). Os dados que foram utilizados para essa função, nesta Tese, são sumarizados na Tabela 6.1.

Tabela 6.1: Tabela com valores dos parâmetros da função *SimDataAffiliation*.

Tipo de Grafo	a	$p1$	$p2$	c	i
C1	$2n$	0.20	0.050	aleatório	0
C2	$2n$	0.25	0.025	aleatório	0
C3	$2n$	0.50	0.005	aleatório	0

Como essa função permite gerar grafos com pesos negativos, a todos os seus valores foi adicionado o valor absoluto do menor peso dentre todas as arestas existentes de forma a torná-las todas não negativas.

A partir dessa função e dos parâmetros apresentados, foram gerados um total de 70 grafos para cada uma das seguintes configurações: aleatória (C1), parcialmente definida (C2), claramente definida (C3). A estrutura aleatória é composta por grafos que não possuem uma estrutura de agrupamento bem definida, o que faz com que os métodos tenham dificuldade para encontrar a partição que gere o melhor agrupamento segundo sua função objetivo. A parcialmente definida é composta por grafos que têm um número um pouco menor de conexões entre *clusters*, se comparados com as conexões intra-*clusters*. Já a última configuração dos grafos, a claramente definida, apresenta alta modularidade, existindo pouquíssimas conexões entre *clusters* e uma quantidade bem maior de conexões intra-*clusters*. Outro parâmetro da função que gera os grafos modulares é o número de *clusters* desejados em tais grafos. Foram gerados grafos com 2, 3, 4, 5, 10, 20 e 50 grupos. Esses grupos não são necessariamente balanceados quanto aos seus números de nós, existindo certa diversidade de tamanhos. Esses tamanhos foram definidos aleatoriamente.

Quanto ao número de nós, foram gerados grafos com 100, 200, 300, 400, 500, 600, 700, 800, 900 e 1000 nós. Para cada valor do número de nós, foram gerados grafos com diferentes números de *clusters*. Por exemplo, foram gerados sete grafos com 100 nós, sendo que cada um deles apresenta 2, 3, 4, 5, 10, 20 e 50 *clusters* em sua estrutura. Foram incluídas ao longo do texto figuras desses grafos simulados, as quais foram criadas por meio da função *Gplot* da biblioteca SIMoNe. Nessas representações dos grafos, os rótulos das classificações obtidas pelos algoritmos estão denotados por cores diferentes. Isso significa que se, por exemplo, o grafo tiver 3 cores diferentes, a sua partição tem 3 *clusters* e nós com mesmas cores pertencem ao mesmo *cluster*.

6.2 Análise da Robustez e Eficiência dos Algoritmos Propostos

Nesta seção, foi realizada uma análise da robustez e da eficiência dos algoritmos propostos. A primeira análise visa verificar se um dado algoritmo, em suas diversas execuções, apresenta resultados parecidos. Como as metaheurísticas propostas possuem uma componente aleatória na sua fase construtiva, diferentes execuções podem resultar em soluções diferentes. O que se busca concluir é que apesar de diferentes, essas soluções não divergem muito de valor.

Para verificar a eficiência dos resultados obtidos pelos métodos propostos, foram realizados dois experimentos: um considerando o valor da sua função objetivo e do seu tempo computacional, e outro considerando seu critério externo utilizando a medida de avaliação *CRand*. No primeiro caso, os resultados das metaheurísticas foram comparados com os de algoritmos encontrados na literatura para resolver os mesmos problemas aqui abordados. A comparação utilizando o critério de avaliação externa considerou todos os algoritmos propostos nesta Tese, que consideraram quatro modelos matemáticos diferentes. Os resultados dessas comparações são apresentados na Seção 6.2.2.

Como se deseja reportar todos os resultados dos testes realizados, ou seja, para os 210 grafos testados, buscou-se um método de apresentação de desempenho que fosse objetivo. Para a análise comparativa dos métodos, decidiu-se adotar o gráfico de desempenho proposto por Dolan e Moré (2002). Esse gráfico foi empregado para comparar os valores das funções objetivo e dos tempos computacionais dos métodos. Para entender como o gráfico de desempenho é elaborado, considere S o conjunto de n_s algoritmos a serem analisados e P o conjunto de n_p problemas teste. Inicialmente, suponha que o problema tenha como objetivo obter o valor mínimo de uma função. Um fator conhecido com raio de desempenho avalia o desempenho do método $s \in S$ aplicado ao problema $p \in P$. O raio de desempenho pode ser calculado pela seguinte equação:

$$r_{ps} = \frac{t_{ps}}{\min\{t_{ps}|s \in S\}}. \tag{6.1}$$

O valor de t_{ps} utilizado na comparação dos métodos informa o desempenho do algoritmo $s \in S$ de acordo com o problema $p \in P$. Segundo Dolan e Moré (2002), quanto mais baixo esse valor, melhor o desempenho do método para o problema. Pela Equação 6.1, pode-se observar que o melhor raio r_{ps} é igual a 1, que ocorre quando t_{ps} corresponde ao valor mínimo. Outro fator considerado por Dolan e Moré (2002) é a seguinte razão: número de problemas para os quais o algoritmo s apresentou um raio de desempenho melhor ou igual ao coeficiente τ dividido pelo número total de problemas do conjunto S. Essa razão pode ser calculada por:

$$\rho_s(\tau) = \frac{1}{n_p}|S(\tau)| \tag{6.2}$$

em que $S(\tau) = \{p \in P|r_{ps} \leq \tau\}$. O valor de $\rho_s(\tau)$ representa a probabilidade de um método s ter um raio de desempenho dentro de um fator τ. No gráfico de desempenho proposto por Dolan e Moré (2002), as curvas são plotadas de acordo com os valores de τ e $\rho_s(\tau)$, para os eixos x e y do gráfico, respectivamente.

Nesta Tese, o conjunto de algoritmos S é composto pelos métodos a serem comparados para as diferentes formulações. Por exemplo, para o problema de maximização de similaridade intra-*cluster*, o conjunto S é composto pelos métodos GRASPI e Ukmeans. Inicialmente, o valor da função objetivo das soluções encontradas para o conjunto P pelos algoritmos do conjunto S é comparado. Nessa comparação, P consiste dos 210 grafos artificiais discutidos anteriormente. Como os problemas aqui estudados são de maximização, para cada função objetivo, o valor do t_{ps} em função do valor da solução foi calculado da seguinte forma:

$$t_{ps} = 1.05 \max\{f_{ps}|s \in S\} - f_{ps}, \tag{6.3}$$

em que f_{ps} é o valor da solução obtida pelo método $s \in S$ aplicado ao problema (ou grafo) $p \in P$. A multiplicação por 1.05 tem como propósito evitar uma divisão por zero.

Um segundo perfil de desempenho, considerando o tempo computacional entre os dois algoritmos do conjunto S é apresentado. Nesse caso, como quanto mais rápido, melhor, foi utilizado o perfil de desempenho como se fosse um problema de mínimo.

6.2.1 Resultado da Análise da Robustez das Metaheurísticas

Para verificar a estabilidade dos resultados encontrados pelas metaheurísticas propostas, elas foram executadas 5 vezes e seus resultados foram comparados. Observou-se que o desvio padrão entre as execuções foi baixo e de difícil análise, então foi utilizada uma alternativa para verificar quão diferentes são as soluções encontradas nas diferentes execuções, avaliando os seus casos extremos. Para cada grafo, manteve-se a sua solução mínima, s_{min}, e máxima, s_{max}, dentre todas as execuções. Avaliou-se o *gap* das soluções encontradas da seguinte forma:

$$gap = \frac{s_{max} - s_{min}}{s_{min}}. \tag{6.4}$$

O *gap* permitirá avaliar a amplitude máxima da diferença das soluções encontradas nas execuções dos algoritmos. Os *gaps* para cada metaheurística foram apresentados dividindo-os em três partes, uma para cada tipo de grafo: C1, C2 e C3. Os resultados obtidos pela metaheurística GRASPI são apresentados nas Figuras 6.1, 6.2 e 6.3. A legenda inferior do eixo-x dessas figuras ilustra o número de *clusters* referentes às partições analisadas e o eixo-y indica o *gap* dos resultados obtidos nas diferentes execuções. A legenda superior do eixo-x indica o número de nós dos grafos particionados referente ao mesmo intervalo no eixo-x inferior.

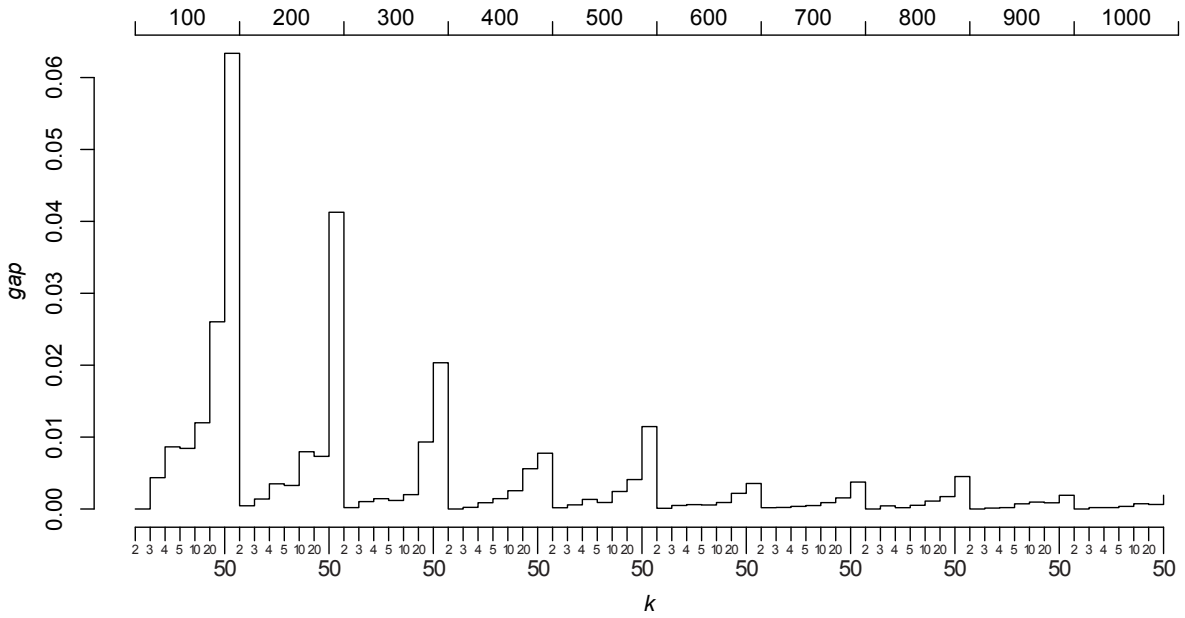


Figura 6.1: Resultados obtidos por GRASPI para grafos do tipo C1.

As Figuras 6.1, 6.2 e 6.3 ilustram o comportamento dos *gaps* do GRASPI para as diferentes classes de grafos modulares gerados, de acordo com os seus números de nós e de

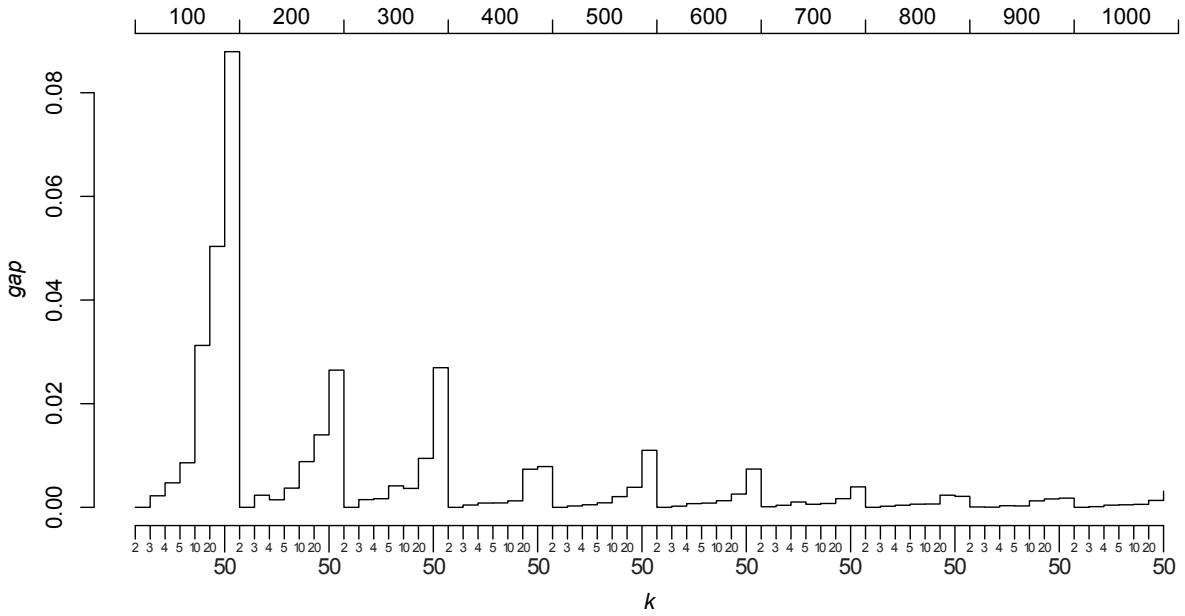


Figura 6.2: Resultados obtidos por GRASPI para grafos do tipo C2.

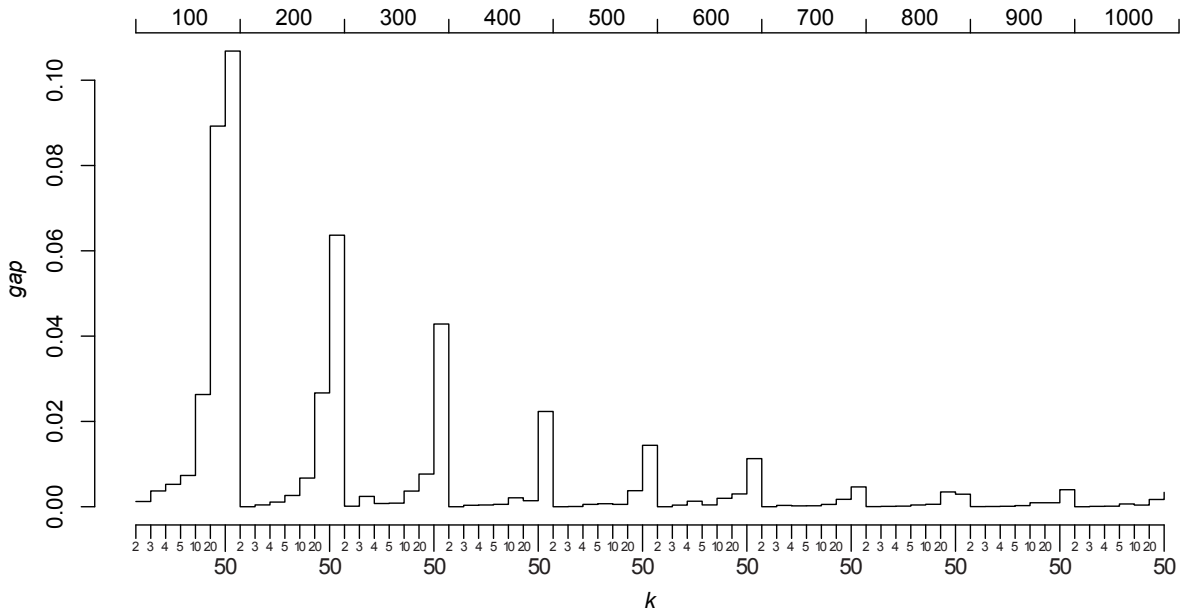


Figura 6.3: Resultados obtidos por GRASPI para grafos do tipo C3.

clusters. Note que o comportamento das curvas dos três gráficos, cada um representando o comportamento dos resultados dos três diferentes tipos de grafos gerados (C1, C2 e C3), apresenta um desenho semelhante. Esse desenho é relativo ao seguinte padrão de estabilidade do GRASPI: o *gap* dos resultados diminui com o aumento do número de nós no grafo, sendo que os resultados apresentaram maior *gap* nos grafos com 100 nós. Independente do número de nós do grafo, quanto menor o seu número de *clusters*, mais estáveis são os resultados da metaheurística, ou seja, os *gaps* aumentam com o aumento do número de *clusters*. Fazendo uma análise comparativa dos três tipos de grafos, nota-se que os maiores picos de *gap* aconteceram de forma gradual, havendo maior *gap* nos grafos com maior estrutura de agrupamento. Entretanto, as diferenças são mais acentuadas nos problemas com menor número de nós e com maior número de *clusters*. De qualquer forma, de maneira geral, a diferença entre os extremos das execuções foi baixa, sendo que o pior caso apresentou pouco mais de 10% de diferença entre as soluções (grafo de 100 nós em 50 grupos).

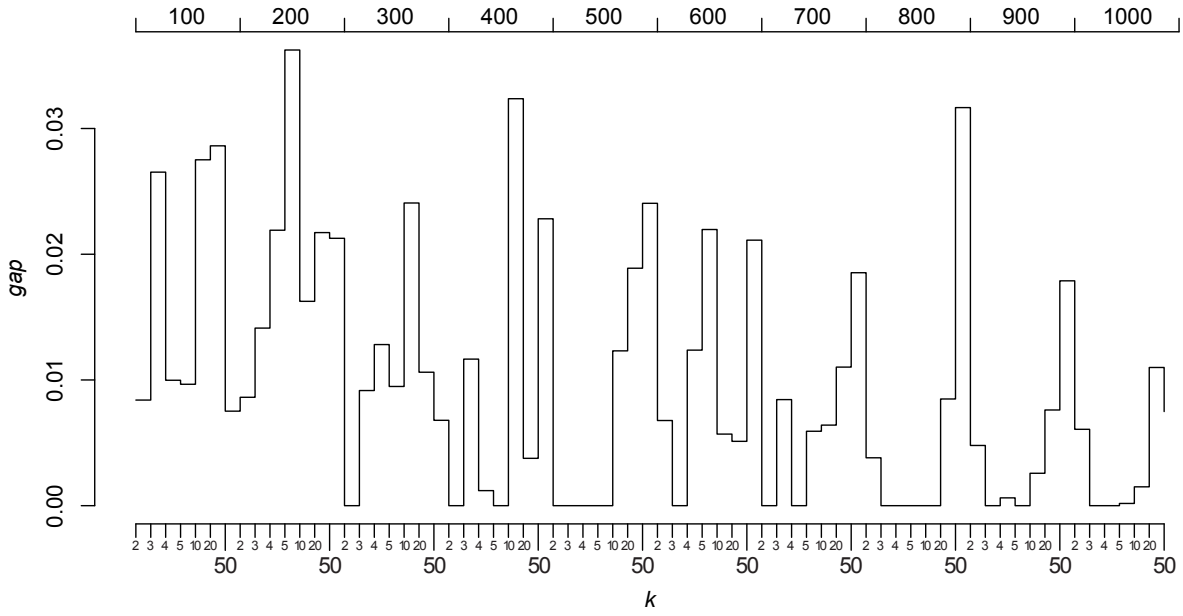


Figura 6.4: Resultados obtidos por GRASPII para grafos do tipo C1.

Pelos resultados observados nas Figuras 6.4, 6.5 e 6.6, nota-se um padrão mais fraco no comportamento dos *gaps* entre as execuções do GRASPII do que entre as do GRASPI. Inicialmente, vale notar que, novamente, o *gap* entre as execuções é baixo, sendo que seus picos acontecem mais freqüentemente quando a partição contém 50 *clusters*. Isso ocorre nos três tipos de grafos. A Figura 6.4 apresenta resultados instáveis entre os diversos *clusters*. Entretanto, seu pico máximo é baixo. Pela Figura 6.5, a diferença de picos entre diferentes números de *clusters* é maior. Por fim, pela Figura 6.6, o comportamento das execuções nos grafos com maior estrutura de agrupamento foi mais padronizado. Há uma irregularidade no padrão no início dessa curva, quando n é baixo. Entretanto, o padrão

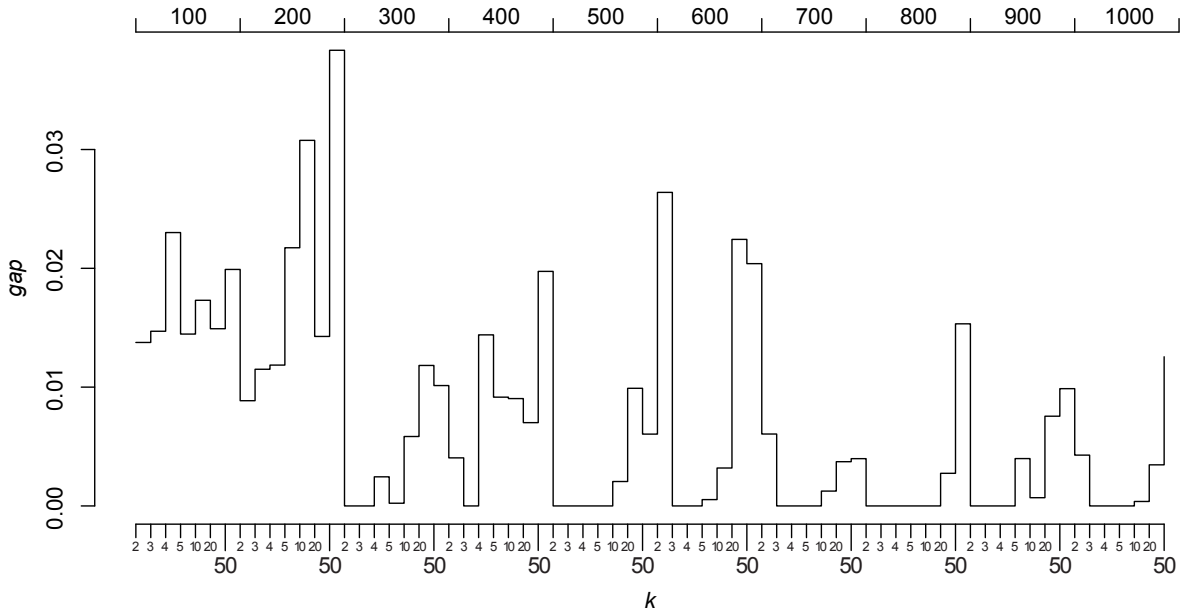


Figura 6.5: Resultados obtidos por GRASPII para grafos do tipo C2.

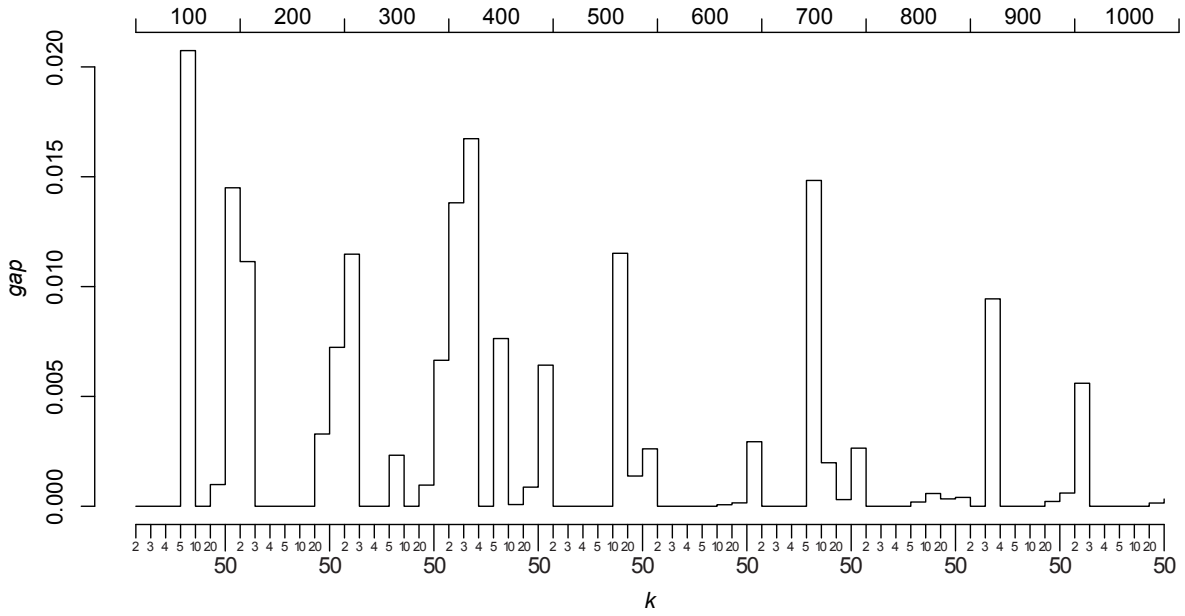


Figura 6.6: Resultados obtidos por GRASPII para grafos do tipo C3.

se torna mais claro com o aumento do número de nós dos grafos particionados, ao mesmo tempo em que os *gaps* vão diminuindo.

6.2.2 Análise Comparativa dos Algoritmos

Para verificar a eficiência das metaheurísticas propostas, elas foram comparadas com algoritmos encontrados na literatura para as mesmas formulações matemáticas estudadas. Cada um desses testes é apresentado em seções separadas para cada formulação estudada: a de maximização da similaridade intra-*cluster* e a de maximização da modularidade das partições. Como a proposta de maximização do *clustering coefficient* de partições é nova, a heurística multi-nível não será comparada com outros algoritmos, por causa da inexistência de algoritmos para a mesma. Contudo, nesta seção, são apresentados os resultados obtidos pelas heurísticas multi-níveis.

Comparação de Métodos para o Modelo de Maximização da Similaridade Intra-Cluster

Esta seção compara o desempenho do GRASPI com o de um algoritmo já existente para o problema de maximização da similaridade intra-*cluster*. O objetivo dessa comparação é validar o método proposto para essa formulação de acordo com seu critério interno, que é a própria função objetivo.

Inicialmente, os algoritmos a serem comparados com o GRASPI seriam o METIS (Karypis e Kumar, 1998) e o Ukmeans (Von Luxburg, 2007), que já foram discutidos na Seção 3.2. Entretanto, a implementação disponibilizada para o primeiro algoritmo pelos seus autores tem como restrição adicional os *clusters* serem balanceados, ou seja, os diferentes *clusters* devem apresentar, aproximadamente, o mesmo número de nós. Por causa dessa restrição, não foi possível comparar o GRASPI com o METIS, pois não seria uma comparação justa, uma vez que o METIS apresentaria soluções para uma formulação diferente. Dessa forma, foi adotado apenas o algoritmo Ukmeans para fazer a análise apresentada.

A Figura 6.7 ilustra o gráfico de desempenho de Dolan e Moré (2002) das soluções do GRASPI e do Ukmeans. De acordo com as curvas correspondentes aos diferentes algoritmos, pode-se observar que o GRASPI obteve o melhor desempenho, independentemente do fator τ. Isso significa que a probabilidade da metaheurística proposta ter um desempenho melhor do que o Ukmeans é sempre maior. O que chama a atenção é a rapidez com que o raio de desempenho do GRASPI atingiu o valor 1. Inicialmente, quando $\tau = 1$, observa-se que o GRASPI tem melhor desempenho em mais de 87% dos problemas, implicando em um resultado geral muito bom. Analisando o restante da curva, pode ser

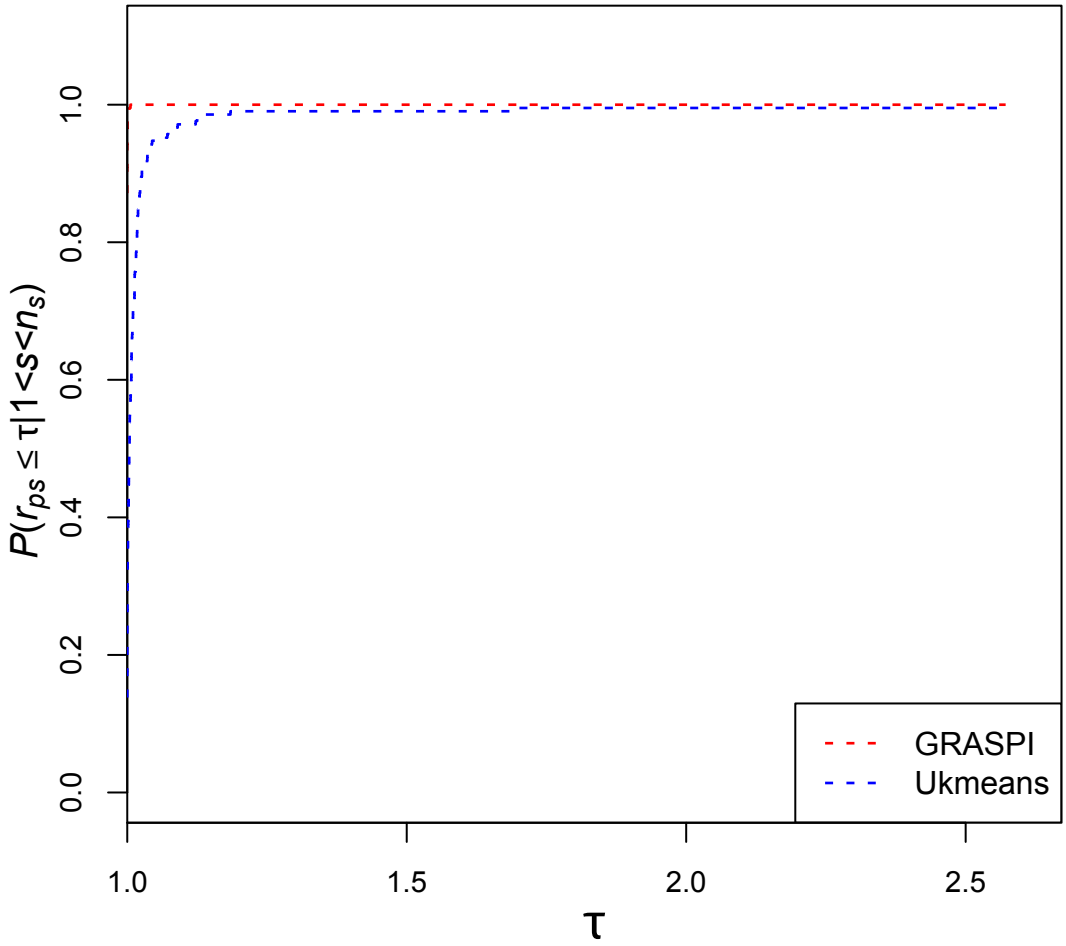


Figura 6.7: Relação gráfica de Dolan e Moré (2002) dos resultados dos algoritmos GRASPI e Ukmeans.

notado que logo em seu início, quando τ é igual a 1.005, o GRASPI já atinge o ponto de máximo. Isso significa que, para os casos em que o GRASPI não é melhor do que o Ukmeans, cerca de 13% dos problemas, o GRASPI apresenta um resultado muito bom, sendo, no pior caso, 0.5% inferior ao Ukmeans.

A Figura 6.8 apresenta a comparação entre os tempos computacionais dos algoritmos GRASPI e Ukmeans segundo o gráfico de perfis de desempenho proposto por Dolan e Moré (2002). Como já era esperado, o tempo computacional do GRASPI foi muito maior do que o do Ukmeans. O Ukmeans, já no início da curva, apontou melhor tempo computacional em todos os seus resultados, enquanto o GRASPI teve no pior caso um desempenho 250 vezes pior. Vale observar que para a maioria dos problemas, mais de 50% (como pode ser observado nesse gráfico) o GRASPI levou no máximo 20 vezes o valor do tempo do Ukmeans para retornar suas soluções. Como os tempos são da ordem de segundos, o tempo real não é tão significativo. Vale observar que o tempo médio do GRASPI foi de 78 segundos, enquanto do Ukmeans foi de quase 2 segundos. Considerando que há

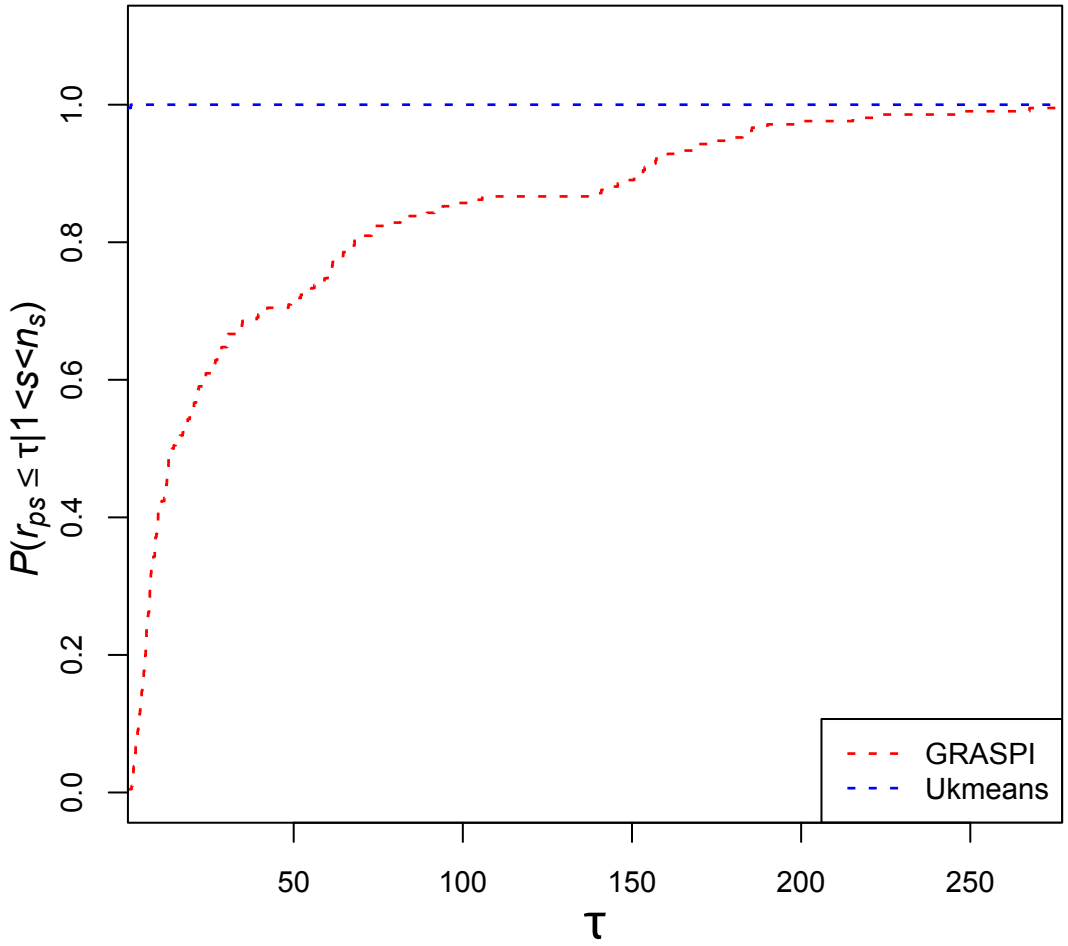


Figura 6.8: Relação gráfica de Dolan e Moré (2002) dos tempos computacionais dos algoritmos GRASPI e Ukmeans.

uma probabilidade maior que 0.9 do GRASPI obter melhor solução do que o Ukmeans, é sugerido esperar o tempo médio de pouco mais de 1 minuto para obter tal solução. No pior caso, para problemas com 1000 nós e 50 *clusters*, o tempo máximo do GRASPI foi de 17.5 minutos, e o do Ukmeans foi de 6 segundos. O tempo mínimo do GRASPI foi de 0.12 segundo, e o do Ukmeans de 0.01 segundo. Uma alternativa para melhorar o tempo computacional do GRASPI é diminuir seu número de iterações, uma vez que isso reduzirá o seu tempo computacional, podendo, no entanto, haver uma piora em sua robustez e na qualidade das soluções.

Com relação ao comportamento das partições, vale observar que houve alguns casos em que o melhor resultado obtido foi para partições com *clusters* compostos por um único nó, como é o caso da partição encontrada para o grafo do tipo C3 com 200 nós e estrutura com 3 *clusters*, ilustrado na Figura 6.9. Esse comportamento da partição encontrada na Figura 6.9 não é desejável em muitas aplicações. Entretanto, quando o grafo a ser particionado era completo, esse tipo de comportamento dos *clusters* das partições não

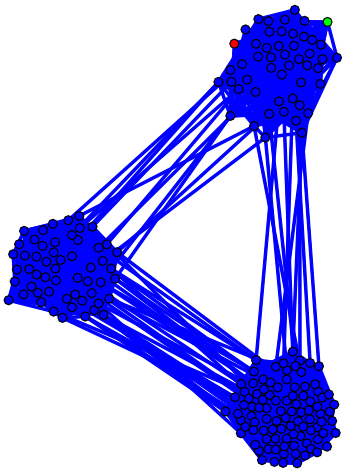


Figura 6.9: Figura ilustrativa da partição resultante do GRASPI do grafo do tipo C3 com 200 nós e estrutura de 3 *clusters*.

foi observado. Pelo contrário, foram atingidos resultados bem satisfatórios para algumas bases biológicas, inclusive apresentando melhores resultados do que alguns modelos muito utilizados na literatura, como o *k-means*. Esses resultados podem ser observados em Nascimento et al. (2010b).

A seguir, esse mesmo tipo de comparação entre algoritmos é realizada usando o GRASPII para o modelo de maximização da modularidade das partições.

Comparação de Algoritmos para o Modelo de Maximização da Modularidade das Partições

Apesar de haver um número considerável de algoritmos propostos para o modelo de maximização da modularidade, poucos são de fácil implementação. Além disso, as implementações disponíveis, em sua maioria, não consideram grafos ponderados. Por essa razão, apenas um algoritmo, o *fast greedy* (Clauset et al., 2004) foi comparado ao GRASPII. A implementação utilizada foi a da biblioteca *igraph* para R-*project*, cuja função é denominada *fastgreedy.community*.

A Figura 6.10 ilustra as curvas de desempenho dos resultados obtidos pelo GRASPII e pelo algoritmo *Fast Greedy*. Pode-se observar que quando $\tau = 1$, que é quando o eixo-y do gráfico fornece a porcentagem do número de vezes em que os algoritmos apresentaram melhor desempenho, o GRASPII atinge aproximadamente 85.3%, enquanto que o algoritmo *Fast Greedy* apresenta pouco mais de 14.7%. Até τ atingir o valor 1.24, o GRASPII alcança um valor melhor do que o *Fast Greedy*, que coincide com o valor de 0.995 do eixo-y. Isso significa que 99.5% dos problemas resolvidos pelo GRASPII obtiveram uma solução no máximo 24% pior do que a melhor solução encontrada pelos dois algoritmos,

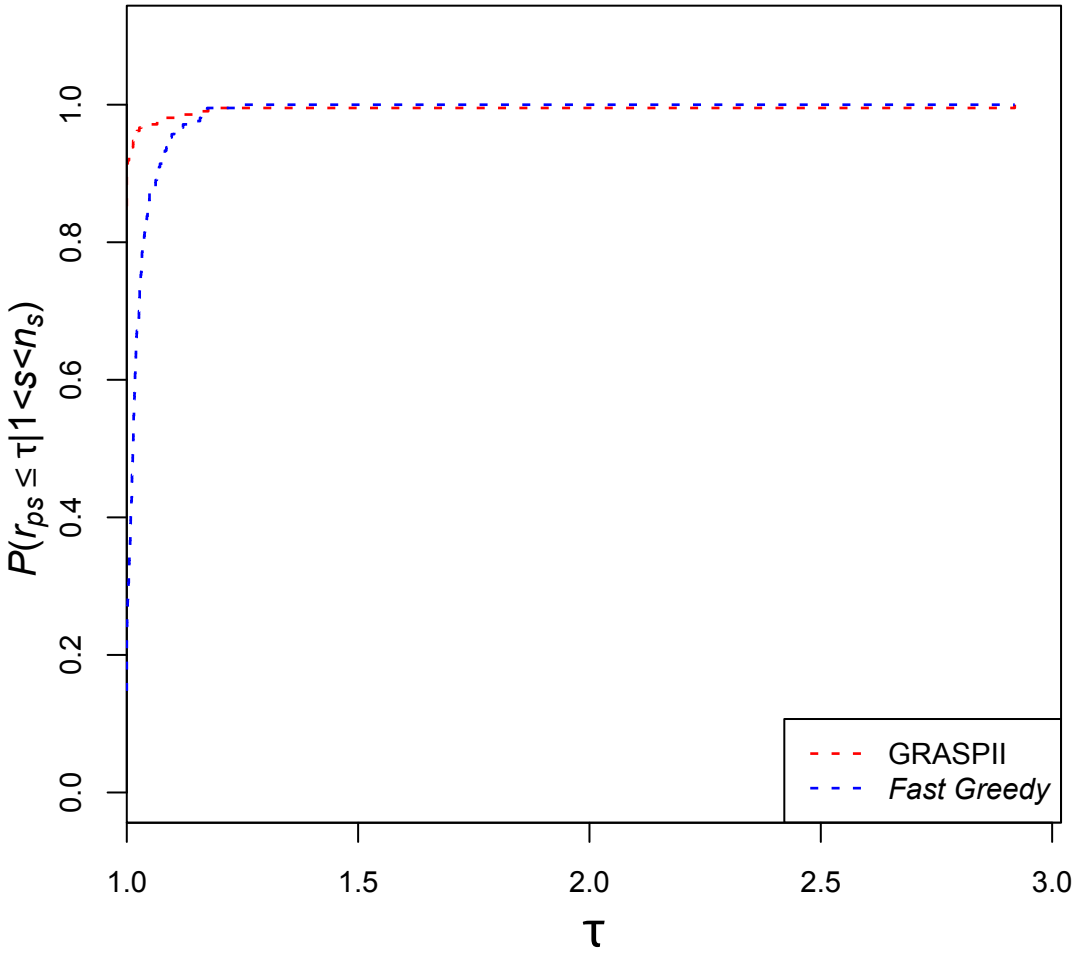


Figura 6.10: Relação gráfica de Dolan e Moré (2002) dos resultados dos algoritmos GRASPII e *Fast Greedy*.

sabendo que em quase 86% desses 99.5%, o valor das soluções obtidas pelo GRASP foi a melhor dentre as obtidas. Entretanto, observou-se que para um problema do conjunto de dados, a solução do GRASPII foi quase 3 vezes pior do que a do *Fast Greedy*, por isso sua demora para convergência para o valor 1.

De acordo com a Figura 6.11, em que são comparados os tempos do GRASPII e do *Fast Greedy*, pode-se observar que o último apresenta o melhor tempo em todos os problemas executados. Nesse caso, nota-se que o GRASPII é muito lento se comparado com o *Fast Greedy*. Vale mencionar que esse último foi especialmente desenvolvido para ser rápido (do inglês *fast*) de forma a poder lidar com grafos formado por milhares de nós. Seu tempo médio é de 1.85 segundos, sendo seu maior tempo de solução igual a 6 segundos, e o mínimo igual a 0.01 segundo. O GRASPII tem um tempo médio de execução de 22 minutos, seu pior tempo de execução é de 1.8 horas e o menor tempo é de 3 segundos.

Alternativas para se melhorar o tempo do GRASPII estão sendo testadas. Notou-se

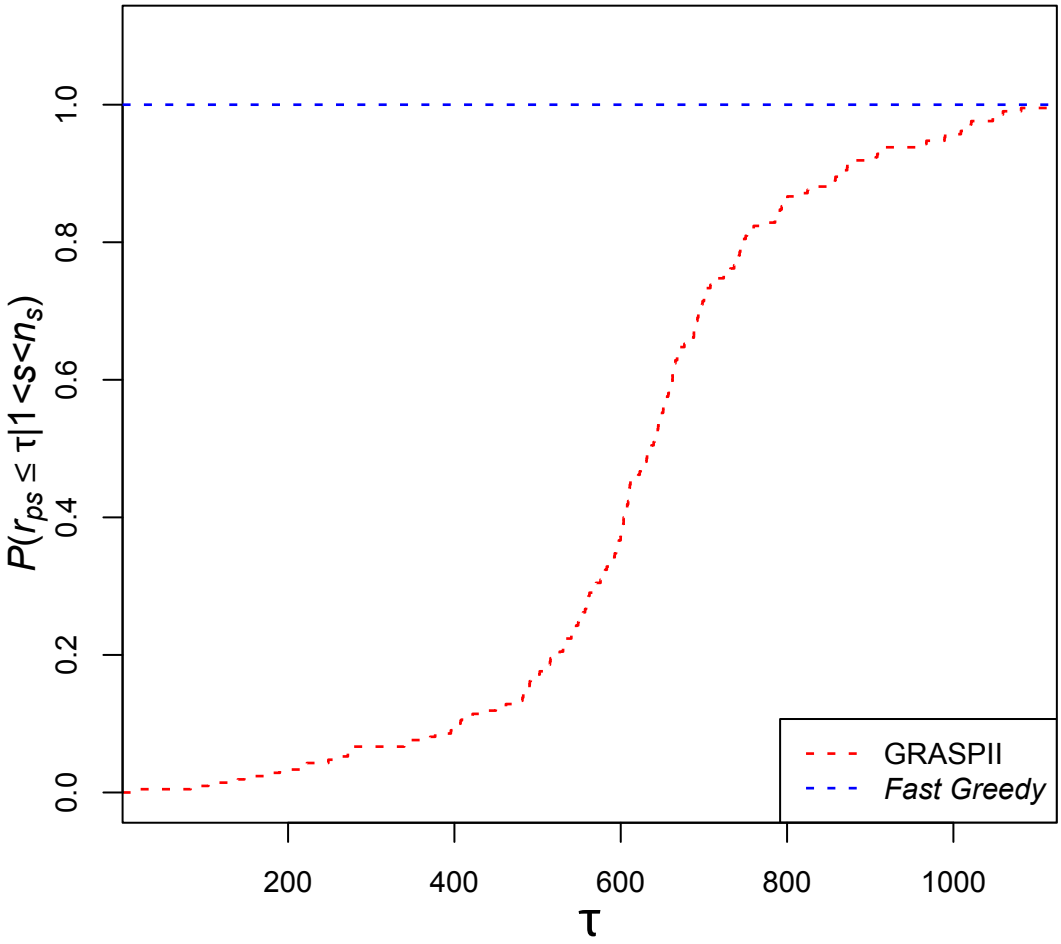


Figura 6.11: Relação gráfica de Dolan e Moré (2002) dos resultados dos algoritmos GRASPII e *Fast Greedy*.

que a fase da metaheurística com maior custo computacional é a construtiva, e maneiras mais eficientes de produzir a lista restrita de candidatos dessa fase, como a utilização de diferentes tipos de algoritmos de ordenação, foram testadas. O algoritmo de ordenação atualmente utilizado é o *heapsort*. Entretanto, devido à quantidade de testes realizados, não há perspectivas de que, com uma pequena modificação no código, esse algoritmo melhore consideravelmente seu tempo de execução. Pode talvez ser necessário readaptá-lo para um algoritmo multi-nível.

Resultados Obtidos para o Modelo de Maximização do Clustering Coefficient das Partições

Esta seção apresenta os resultados obtidos pelos algoritmos multi-níveis propostos. Os testes foram restringidos a grafos ponderados, o alvo de estudo desta Tese.

Primeira Proposta de Clustering Coefficient

Testes preliminares utilizando dados simulados apontaram que os *clusters* das parti-

ções obtidas pelo algoritmo CCMLI em grafos do tipo C3 (com clara definição de grupos e que não eram grafos completos e com pesos) apresentaram *clusters* com um número de nós muito baixo, como ilustrado na Figura 6.12.

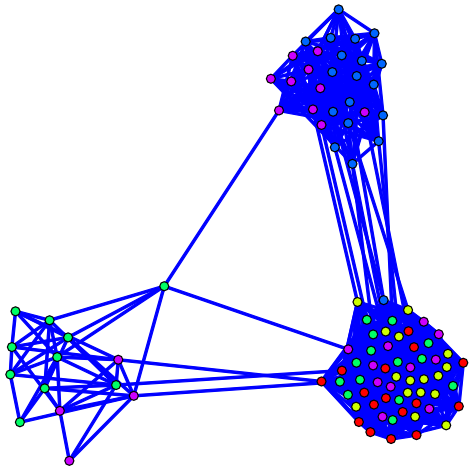


Figura 6.12: Figura ilustrativa da partição resultante do CCMLI do grafo do tipo C3 com 100 nós e estrutura de 3 *clusters*.

Investigando as possíveis causas de tal evento, descobriu-se que, mesmo tendo os pesos normalizados, a consideração do clique em sua razão fazia com que apenas os nós que formavam triângulos com intensidade alta fossem inseridos em um mesmo *cluster*, havendo uma fragmentação elevada dos grupos para a melhoria do valor do *clustering coefficient* proposto. Além disso, é possível observar na mesma figura que nós não adjacentes eram inseridos em um mesmo *cluster* com freqüência. Isso se deve à indiferença no valor da função objetivo quando há adição de nós não adjacentes em um mesmo *cluster*, já que o grau dos nós de cada *cluster* continua o mesmo após a sua união (pois não há relação de adjacência entre seus nós). Esse evento é bastante indesejado, pois as partições podem gerar grupos não condizentes com a conectividade dos nós.

Continuando o estudo de caso, foi observado que, para grafos completos e ponderados, o CCMLI foi capaz de detectar corretamente os grupos do grafo. Logo, buscou-se um pré-processamento dos grafos que não fossem completos de forma a torná-los completos, resultando em uma matriz de pesos em que todos os seus elementos, exceto a sua diagonal, tenham valores maiores do que zero. Esse pré-processamento se resume a, dado um grafo que não é completo, inicialmente considerar uma matriz de dissimilaridades $D = [d_{ij}]_{n\times n}$, em que $d_{ij} = 1.01 \max\{w_{ij} | i,j \in \mathbb{Z}, 1 \leq i,j \leq n\} - w_{ij}$. Posteriormente, calcula-se a matriz dos caminhos mais próximos $T = [t_{ij}]_{n\times n}$ por meio dessa matriz D, e então se considera a matriz de similaridades $S = [s_{ij}]_{n\times n}$, em que $s_{ij} = 1 - t_{ij} / \max\{t_{ij} | i,j \in \mathbb{Z}, 1 \leq i,j \leq n\}$. Utilizando essa matriz de similaridades como uma matriz de pesos, as partições encontradas apresentaram maior coerência, como era esperado. Para o exemplo da Figura 6.12, o

método obteve a partição apresentada na Figura 6.13.

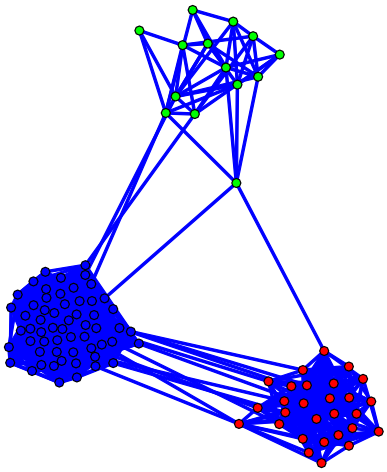


Figura 6.13: Figura ilustrativa da partição resultante do CCMLI utilizando a matriz dos caminhos mais próximos do grafo do tipo C3 com 100 nós e estrutura de 3 *clusters*.

Mesmo com esse pré-processamento, algumas partições apresentaram ramificações intra-*cluster* indesejadas. Foi observado então que, conforme o *cluster* apresentava um número maior de nós, o CCMLI tendia a ramificar os seus *clusters* para aumentar a função objetivo. Esse evento foi observado no grafo da Figura 6.14.

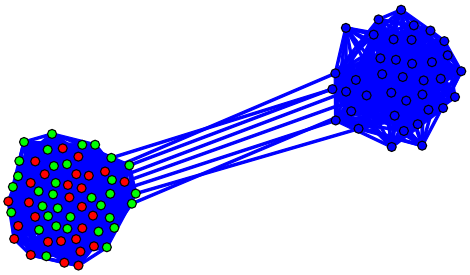


Figura 6.14: Figura ilustrativa da partição resultante do CCMLI utilizando a matriz dos caminhos mais próximos do grafo do tipo C3 com 100 nós e estrutura de 2 *clusters*.

Além de o problema mencionado ter sido resolvido apenas parcialmente, uma limitação associada a essa estratégia de pré-processamento é o custo de se manipular um grafo completo. Quando o número de nós é superior a 500, o tempo médio de execução aumenta significativamente, sendo necessário aproximadamente 20 minutos para encontrar uma solução por meio dessa heurística. Dessa forma, decidiu-se analisar as partições sem a conversão para a matriz dos caminhos mais próximos. Os valores das soluções médias encontradas para cada tipo e tamanho de grafo são apresentadas na Tabela 6.2.

De acordo com a Tabela 6.2, pode ser observado que as soluções médias dos grafos

Tabela 6.2: Tabela com valores das soluções médias encontradas pelo algoritmo CCMLI.

Tipo de	n									
Grafo	100	200	300	400	500	600	700	800	900	1000
C1	0.177	0.223	0.229	0.219	0.100	0.089	0.088	0.090	0.090	0.092
C2	0.219	0.231	0.229	0.245	0.124	0.119	0.126	0.122	0.124	0.116
C3	0.257	0.384	0.307	0.333	0.249	0.257	0.253	0.249	0.252	0.251

crescem conforme a estrutura do grafo se torna mais definida. Esses resultados sugerem que grafos com maior tendência de agrupamento têm um maior valor de solução, o que já era esperado devido às características da medida.

A seguir, os resultados da outra proposta de formulação para agrupamento em grafos são apresentados.

Segunda Proposta de Clustering Coefficient

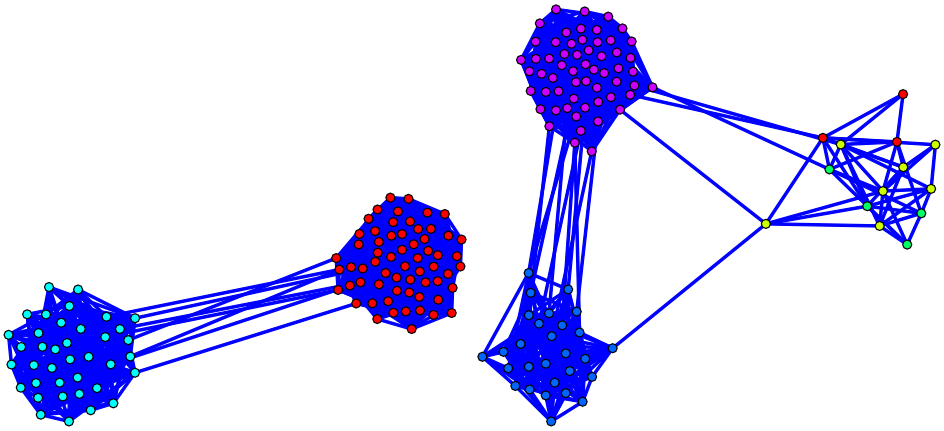
Ao estudar as partições encontradas heuristicamente pela proposta de formulação de agrupamento em grafos baseada no *clustering coefficient* proposto por Barrat et al. (2004), verificou-se o surgimento de partições mais condizentes com a divisão natural dos dados. A heurística multi-nível, CCMLII, obteve resultados satisfatórios, havendo um índice de acerto elevado com relação às classes dos grafos Gaussianos gerados aleatoriamente, verificável visualmente pelos exemplos das representações gráficas dos resultados apresentados na Figura 6.15, e ratificados na Seção 6.2.3. Nessa figura (considere os rótulos das partições dos nós como as diferentes cores) pode ser observado que para diferentes grafos com estrutura de agrupamento, a detecção dos grupos foi bem satisfatória.

Tabela 6.3: Tabela com valores das soluções médias encontradas pelo algoritmo CCMLII.

Tipo de	n									
Grafo	100	200	300	400	500	600	700	800	900	1000
C1	0.180	0.151	0.134	0.121	0.113	0.118	0.114	0.112	0.112	0.110
C2	0.198	0.184	0.176	0.185	0.145	0.154	0.150	0.153	0.156	0.156
C3	0.398	0.430	0.450	0.454	0.452	0.456	0.452	0.453	0.457	0.457

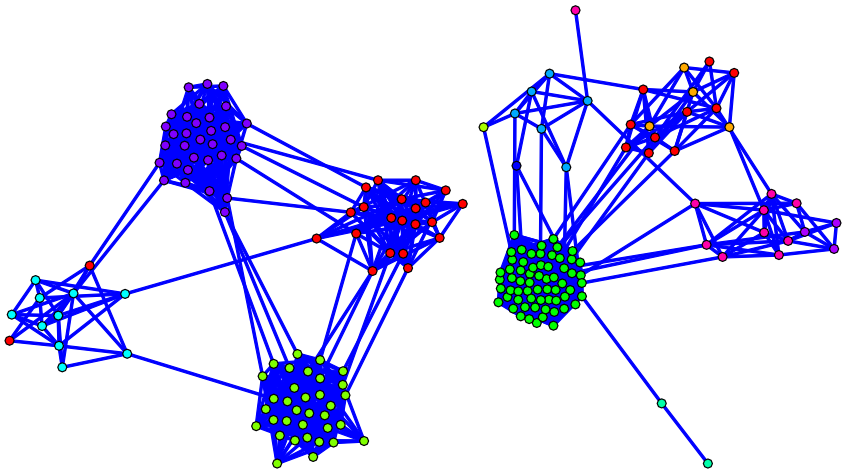
A Tabela 6.3 apresenta os valores das soluções médias para cada tipo e tamanho de grafo dos testes computacionais. Pode-se verificar que, como esperado, as soluções médias obtidas para os grafos com uma estrutura de agrupamento mais definida foram mais altas do que para os grafos de outros tipos. Além disso, não foi observado nenhum padrão com o aumento ou a diminuição do número de nós do grafo. Levando em consideração a média das soluções do grafo do tipo C3, observou-se uma estabilidade nos valores médios das soluções.

A vantagem das formulações propostas é que, além de fornecerem partições baseadas na conectividade do grafo, os valores obtidos pelas funções objetivo fornecem a tendência



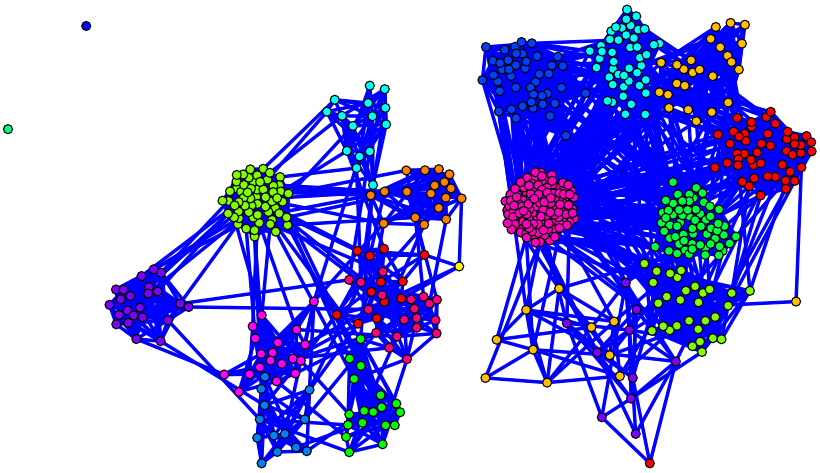
(a) Grafo com 100 nós e 2 *clusters*

(b) Grafo com 100 nós 3 *clusters*



(c) Grafo com 100 nós e 4 *clusters*

(d) Grafo com 100 nós e 5 *clusters*



(e) Grafo com 200 nós e 10 *clusters*

(f) Grafo com 400 nós e 10 *clusters*

Figura 6.15: Estes grafos ilustram as partições encontradas pela formulação baseada no *clustering coefficient* de Barrat et al. (2004).

de agrupamento das partições. As funções objetivo permitem inferir importantes informações sobre os grafos. Por exemplo, como pôde ser observado nos resultados da Tabela 6.3, se o grafo tem uma estrutura de agrupamento não muito bem definida, como é o caso dos grafos do tipo C1, os valores das funções objetivo das partições encontradas são inferiores. Essa informação é bastante útil na área de aprendizado não supervisionado, pois muitas vezes os dados não podem ser visualizados, por falta de ferramentas, ou devido às suas dimensões, e extrair uma informação numérica do tipo de agrupamento obtido é de clara relevância. Dessa forma, as formulações propostas baseadas na medida de *clustering coefficient* permitem encontrar partições de boa qualidade, sendo a qualidade dessas partições relativa à estrutura do grafo estudado.

6.2.3 Avaliação Externa das Partições

Nesta seção, as partições encontradas por todos os algoritmos propostos são avaliadas segundo o critério de avaliação externa *CRand*. Essa avaliação tem como objetivo estudar a validade das partições encontradas segundo a classificação atribuída aos dados gerados. Vale ressaltar que a medida mais relevante de avaliação da qualidade das soluções encontradas em cada algoritmo é a própria função objetivo. Entretanto, o critério de avaliação externa tem um papel importante nesta Tese, pois como foram propostos modelos de agrupamento, é necessário avaliar se eles são coerentes ao fornecer partições para o problema de agrupamento em grafos. Nesses testes, apenas o conjunto de grafos do tipo C3 foi considerado, por se julgar o tipo de agrupamento e classes atribuídas na geração desses grafos mais condizentes com a estrutura de grafos para agrupamento. A primeira comparação a ser realizada avalia todos os algoritmos propostos nesta Tese com relação ao critério *CRand*. Nesse caso, são considerados os algoritmos GRASPI, GRASPII, CCMLI e CCMLII.

A Figura 6.16 apresenta o gráfico de perfis de desempenho comparando os quatro algoritmos propostos nesta Tese. Os algoritmos CCMLI e GRASPI foram os que apresentaram os piores desempenhos. Para $\tau = 1$, ambos encontram a melhor solução dentre todos os algoritmos em apenas um grafo, o que corresponde a 1.43% do conjunto de dados. Além disso, nos piores casos, que corresponde a aproximadamente 20% dos problemas testados, os resultados obtidos por esses algoritmos são 100 vezes pior do que o melhor resultado obtido comparando todos os algoritmos. Como o CCMLII apresentou melhores resultados do que o CCMLI com relação à classificação dos dados, motivo já discutido na Seção 6.2.2, decidiu-se realizar outra análise comparativa, agora considerando apenas a formulação proposta do *clustering coefficient* que deu indícios de melhor desempenho dentre os modelos propostos nesta Tese (o baseado na medida de Barrat et al. (2004)) e

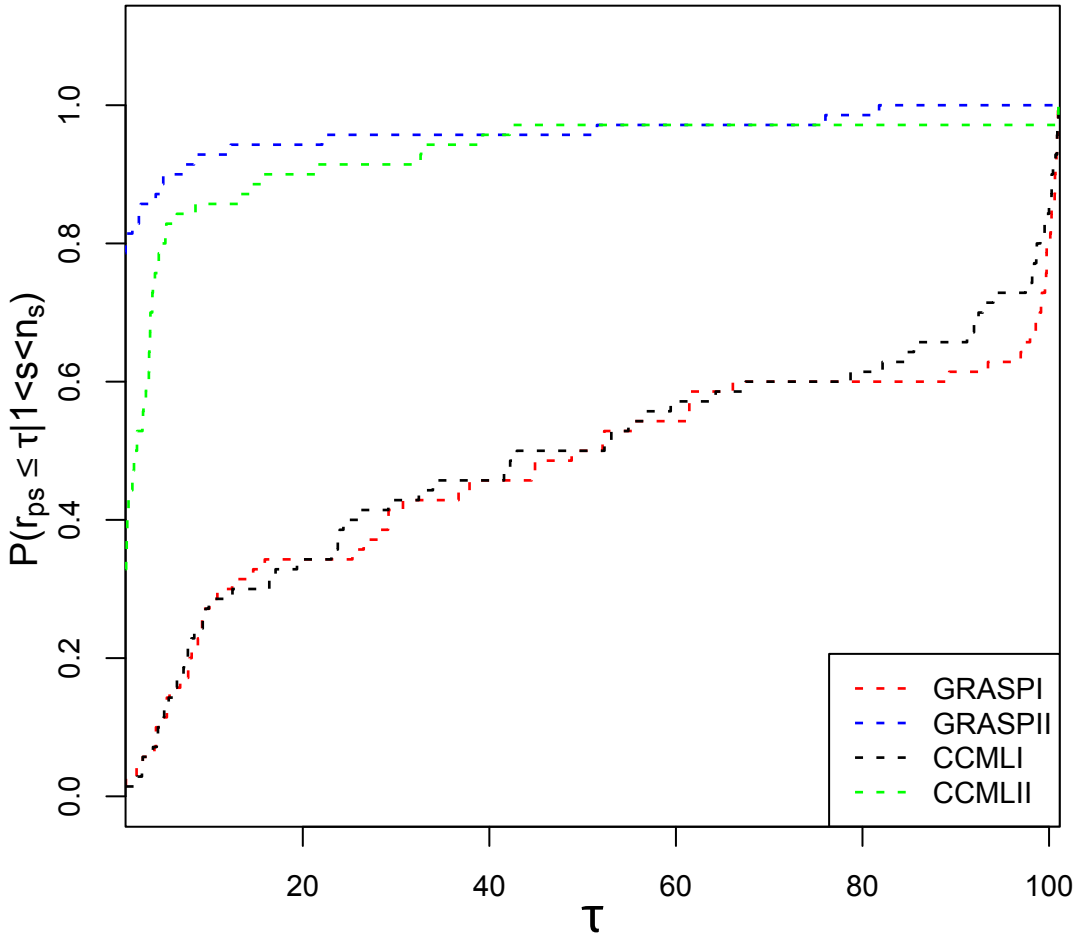


Figura 6.16: Gráfico ilustrativa de todos os algoritmos propostos com relação ao índice *CRand*. As partições consideradas são as resultantes dos grafos do tipo C3.

a medida atualmente mais usada na solução do problema de detecção de comunidades, que é a baseada na maximização da modularidade das partições. Para fazer essa análise, foi necessário intensificar a busca local do algoritmo multi-nível CCMLII. A intenção era encontrar resultados de melhor qualidade para poder fazer inferências sobre as partições encontradas pelo modelo proposto. Dessa forma, a busca local da fase de refinamento foi adaptada usando a busca local proposta para o GRASPI e GRASPII. Nessa adaptação, para executar a transferência de objetos entre *clusters*, é necessário levar em consideração a melhoria da solução sem limitar sua vizinhança, como acontece na fase de refinamento do CCMLII. Como resultado dessa modificação, houve um grande aumento no custo computacional, pois a avaliação dos movimentos de transferência de objetos entre *clusters* é não-linear, difícil de se avaliar por meio de uma equação que não considere todos os objetos envolvidos no par de *clusters* envolvido na transferência. As partições que obtiveram maior modularidade dentre os algoritmos GRASPII e *Fast Greedy* foram consideradas nessa comparação.

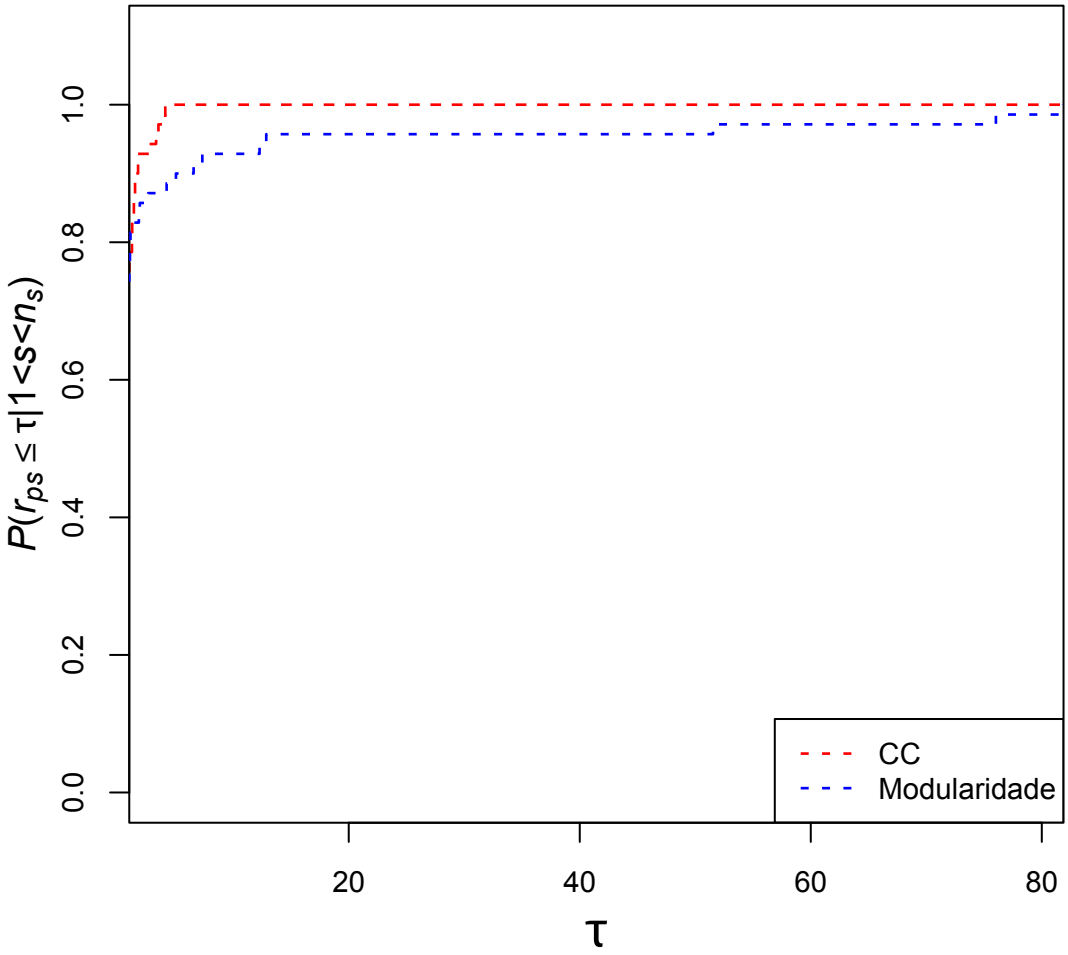


Figura 6.17: Figura que apresenta o desempenho da avaliação externa das partições encontradas pela formulação de maximização *clustering coefficient* baseada no índice de Barrat et al. (2004) e pela formulação de maximização da modularidade dos 70 grafos do tipo C3.

A Figura 6.17 apresenta o gráfico de desempenho de Dolan e Moré (2002) considerando os resultados obtidos pela maximização do *clustering coefficient* (CC) baseado no índice de Barrat et al. (2004) e pelo modelo de maximização da modularidade das partições. Nesta avaliação, o que se compara é o *CRand* das partições encontradas pelos dois modelos. Na Figura 6.17, considerando o fator $\tau = 1$, pode-se observar que o modelo proposto e o de maximização de modularidade apresentaram resultados bem competitivos, sendo que o primeiro apresentou um valor 0.757 no eixo-y, enquanto o segundo obteve 0.743. Isso significa que o modelo proposto apresentou o maior índice *CRand* em 75.7% dos problemas, enquanto que o de maximização da modularidade apresentou para 74.3% dos problemas. Portanto, se o leitor fosse considerar apenas o índice de acerto dos modelos, ou seja, o número de vezes em que um modelo apresentou partições com maior *CRand*, a conclusão seria que os dois modelos foram bem competitivos, tendo o

aqui proposto um desempenho ligeiramente melhor. Entretanto, analisando o restante da curva de desempenho, pode-se observar que o modelo proposto apresentou o melhor desempenho para quase todos os valores de τ, exceto no intervalo [1.002,1.24], em que o modelo de maximização da modularidade apresenta uma probabilidade ligeiramente melhor do que o proposto, sendo a maior diferença nesse intervalo de 2 pontos percentuais. A partir do $\tau = 1.24$, observa-se uma alta taxa de crescimento na curva correspondente ao modelo proposto, que atinge seu máximo (eixo-y igual a 1) quando $\tau = 4.12$. Nesse ponto, a curva correspondente à maximização da modularidade atinge 0.87 no eixo y, apresentando seu valor máximo apenas quando $\tau = 81$. O que pode se inferir a partir desse resultado é que, considerando os resultados até 24% piores que o melhor atingido, ambos os modelos apresentam aproximadamente o mesmo comportamento. Entretanto, como pode ser observado na Figura 6.17, o modelo de maximização de modularidade teve um conjunto de 13% de seus problemas (9 de 70) com o *CRand* pior do que o do modelo proposto. O pior caso foi o de um problema com um *CRand* 80 vezes pior do que o do modelo proposto, e o melhor com *CRand* 5 vezes pior.

Dessa forma, pode-se concluir que o modelo proposto baseado na medida de Barrat et al. (2004) apresentou resultados bastante consistentes em um conjunto de 70 grafos de diversos tamanhos e número de *clusters* naturais. Além disso, os resultados apontaram melhor estabilidade das partições encontradas por esse modelo, quando comparado a um modelo muito utilizado na última década, conhecido como modelo de maximização da modularidade das partições.

A seguir, são apresentados alguns experimentos utilizando bases de dados reais extraídas da literatura. Nesses experimentos, as duas formulações estudadas nesta Tese e as duas propostas são avaliadas. Os seus resultados são confrontados utilizando a medida de avaliação externa *CRand*.

6.3 Testes com Dados Reais

Esta seção apresenta os agrupamentos encontrados para conjuntos de dados reais. A maioria desses dados é usada em problemas de classificação de dados e a aplicação dos algoritmos propostos nesta Tese a tais problemas visa apenas identificar a qualidade dos agrupamentos encontrados com relação ao rótulo dos dados reais. Para isso, a medida de avaliação utilizada é o *CRand*. Os dados utilizados nesta Tese são apresentados a seguir.

6.3.1 Conjunto de Dados

O primeiro conjunto de dados a ser apresentado é o Golub (Golub et al., 1999), que consiste de 72 objetos de tecidos com dois tipos distintos de leucemia linfocítica aguda (ALL- *acute lymphoblastic leukemia*), com 47 objetos, e leucemia mielóide aguda (AML - *acute myeloid leukemia*), com 25 objetos. Esses objetos são formados por níveis de expressão gênica obtidos por meio de *microarrays* de DNA. De acordo com Golub et al. (1999), categorizar ALL e AML é importante para o sucesso do tratamento desses tipos de leucemias, e nenhum teste existente é adequado para o seu diagnóstico. Embora o tratamento usado para ALL possa ser utilizado no tratamento de pacientes com AML, e vice-versa, o tratamento específico leva a uma melhor taxa de cura. Por essa razão, a distinção entre os dois tipos de leucemia aguda merece ser estudada.

Golub et al. (1999) propuseram uma abordagem para distinguir os tipos de leucemia e, por meio de observações, os autores decompuseram ALL em duas classes: linhagem-B (B-ALL) e linhagem-T (T-ALL). Nos conjuntos de dados, 38 de 47 objetos da leucemia ALL correspondem à linhagem-B, enquanto 9 correspondem à linhagem-T. Dessa forma, o conjunto de dados Golub pode ser analisado de acordo com esses 3 subtipos de leucemia (AML, T-ALL e B-ALL). O conjunto de dados original tem 6817 níveis de expressão gênica, ao qual Dudoit et al. (2000) aplicaram um pré-processamento que resultou na redução para 3571 genes. O conjunto de dados utilizado nesta Tese foi a versão pré-processada. Para mais detalhes do pré-processamento, consulte Dudoit et al. (2000) e, para mais detalhes do conjunto de dados original, veja Golub et al. (1999).

O segundo conjunto de dados é o Leukemia, composto por 327 objetos de ALL pediátrico da medula espinhal. Esse conjunto de dados foi estudado por Yeoh et al. (2002) obtendo seus objetos por análise *microarray*. O ALL consiste de diferentes subtipos de leucemia e os objetos podem ser divididos em 6 classes: BCR-ABL (15 objetos), E2A-PBX1 (27 objetos), Hiperdiplóide>50 (64 objetos), MLL (20 objetos), T-ALL (43 objetos) e TEL-AML1 (79 objetos). Além disso, há um grupo de objetos que não pertencem a nenhum subtipo descrito anteriormente (79 objetos), totalizando 7 classes no conjunto de dados. O conjunto de dados original tem 12558 genes, no entanto, para análise dos dados, foi utilizada uma versão pré-processada por Yeoh et al. (2002) com 271 genes.

Breast (Bennett e Mangasarian, 1992) é uma base de dados de células cancerosas que possui 699 objetos compostos por 9 atributos. A classificação dessas células é dada segundo a sua natureza, como maligna ou benigna, confirmada pela técnica *fine needle aspirate* (FNA). FNA permite a biópsia por tecidos recuperados por meio de análise microscópica. Os atributos desse conjunto de dados são: a espessura do grupo, a uniformidade do tamanho da célula, a uniformidade do formato da célula, a adesão marginal,

o tamanho da célula epitelial única, o núcleo vazio, a cromatina suave, o nucléolo normal e a mitose. O número de objetos de natureza benigna é igual a 241, enquanto que 458 objetos são da classe maligna.

Novartis (Su et al., 2002; Monti et al., 2003) é uma base de dados de expressão gênica com 103 objetos de tecidos cancerosos. Esses objetos são de 4 diferentes tipos de câncer: 26 de mama, 26 de próstata, 28 de pulmão e 23 de cólon. Cada objeto tem 1000 atributos, que correspondem a níveis de expressão de 1000 genes. O conjunto de dados MultiA (Su et al., 2002) tem os mesmos objetos, só que pré-processadas por Hoshida et al. (2007), e com um número maior de atributos, 5565. Para manipulação desse conjunto de dados, essas bases precisam ser normalizadas devido à não uniformidade dos dados e dos diferentes limites nos valores dos atributos.

O conjunto de dados MultiB (Ramaswamy et al., 2001) é, assim como MultiA e Novartis, um conjunto de dados com tecidos de células tumorais. Esse conjunto de dados, pré-processado por Hoshida et al. (2007) para reduzir sua dimensionalidade, ficou com 5565 atributos. O conjunto de dados, que possui 32 objetos, pode ser classificado de acordo com seus tipos de tecidos: 5 correspondem a tecidos de mama, 9 de próstata, 7 de pulmão e 11 de cólon.

Lung (Bhattacharjee et al., 2001) é formado por dados que representam níveis de expressão do RNAm de tumores de pulmão coletados por *microarrays* de oligonucleotídio. Essa base contém 17 objetos de tecido normal e 3 tipos de câncer de pulmão, correspondendo a 139 objetos de adenocarcinomas, 21 de carcinomas de células escamosas e 20 de carcinóides. Nos testes realizados, foi usada uma versão desses dados proposta por Monti et al. (2003).

MiRNA (Lu et al., 2005) é uma base de dados de uma pequena classe não codificada de espécies de RNA, os microRNAs (miRNAs). Os dados são obtidos por meio de um método de coleta de perfil baseado na expressão de 217 miRNAs de mamíferos. Essa base contém 218 objetos de tecidos humanos de diferentes tipos de tumores. Os objetos podem ser classificados em:

- epiteliais (83 objetos) e não epiteliais (135 objetos), de acordo com a origem do objeto;
- gastrointestinais (184 objetos) e não gastrointestinais (34 objetos), também de acordo com a origem dos objetos;
- normais (46 objetos), tumores (140 objetos) e linhagem de células (32 objetos), quanto à natureza dos objetos;

- estomacais (6 objetos), de cólon (15 objetos), pancreáticas (10 objetos), renais (11 objetos), hepáticas (3 objetos), de bexiga (9 objetos), de próstata (18 objetos), de ovário (7 objetos), de útero (19 objetos), pulmonar (10 objetos), de mesotelioma (8 objetos), de melanoma (5 objetos), de mama (14 objetos), de cérebro (2 objetos), de B-ALL (26 objetos), de T-ALL (28 objetos), de células B de linfoma grandes (8 objetos), de linfoma folicular com fendas (8 objetos), de micose fungóide (3 objetos) e de AML (8 objetos), definindo o tipo de tecido;
- 8 tipos diferentes de 3, 4, 2, 5, 10, 3, 2 e 3 objetos definindo a linhagem de células, originando 9 classes (incluindo uma classe de objetos que não são de linhagens de células, correspondendo a 186 objetos);
- 4 classes, por meio da informação da origem das objetos, que foram colhidas para o estudo de Ramaswamy (123 objetos), no hospital St Jude (31 objetos), no hospital Dana-Farber (39 objetos) e no MIT (25 objetos).

Proteínas (Ding e Dubchak, 2001) é um conjunto de dados composto por 698 objetos correspondentes a dobras de proteínas, com 125 atributos cada. O reconhecimento de dobras de proteínas é importante para o entendimento de sua estrutura, sem precisar se preocupar com a similaridade da seqüência. Essa base de dados pode ser classificada em duas estruturas diferentes. A primeira estrutura é relativa a 4 grandes classes de dobras principais com 116, 226, 260 e 96 objetos cada uma, e a segunda representa uma divisão em 27 subtipos de dobras com 19, 16, 32, 15, 18, 16, 74, 21, 29, 13, 16, 32, 12, 13, 16, 77, 23, 24, 40, 22, 17, 24, 18, 15, 15, 40 e 41 objetos.

Yeast (Nakai e Kanehisa, 1991) é uma base de dados com objetos correspondentes a 1484 proteínas de levedura com 8 atributos relativos às características calculadas a partir das seqüências de aminoácidos. O sítio de localização de uma proteína dentro de uma célula é primordialmente determinado pela seqüência do aminoácido. As proteínas da Yeast podem ser classificadas em 10 tipos diferentes, levando-se em consideração o sítio de localização da proteína: citoplasmática ou citoesqueleto (463 objetos), nuclear (429 objetos), mitocondrial (244 objetos), proteína de membrana sem sinal N-terminal (163 objetos), proteína de membrana sem mórula com sinal de divisão (51 objetos), proteína de membrana com mórula com sinal de divisão (44 objetos), extracelular (37 objetos), vacuolar (30 objetos), peroxissomal (20 objetos), e localizadas no lúmen do retículo endoplasmático (5 objetos).

Ecoli (Nakai e Kanehisa, 1991) é uma base de dados com 336 objetos de proteínas da bactéria *Escherichia coli*. A partir das seqüências de aminoácidos dessas proteínas, 7 atributos foram utilizados para a classificação dos dados. Essas proteínas podem ser classificadas em 8 tipos diferentes levando-se em consideração o sítio de localização da

proteína: citoplasmática, com 143 objetos; membrana interna sem seqüência de sinal, com 73 objetos; periplasma, com 52 objetos; membrana interna com sinal de seqüência de divisão, com 35 objetos; membrana externa, com 20 objetos; membrana externa lipoproteica, com 5 objetos; membrana interna, com 2 objetos; e membrana interna com divisão de seqüência de sinal, com 2 objetos.

BreastA (van 't Veer et al., 2002) e BreastB (West et al., 2001) são dois conjuntos de dados com, respectivamente, 98 e 48 objetos de tumor de mama obtidos usando *microarray* de oligonucleotídeos com um e com dois canais. Os conjuntos de dados utilizados nesta Tese são pré-processados e cada um deles apresenta 1213 atributos. Com relação à classificação, Hoshida et al. (2007) apresentaram uma análise do conjunto de dados BreastA dividindo-o em 3 classes com 11, 51 e 36 objetos. BreastB, assim como em (Nascimento et al., 2010b), foi decomposta em 2 classes levando-se em consideração o receptor de estrogênio (RE). Esse conjunto de dados tem 25 objetos de RE positivo (RE+) e 24 objetos de ER negativo (RE-).

Os conjuntos de dados DLBCLA (Monti et al., 2005), DLBCLB (Rosenwald et al., 2002) e DLBCLC Shipp et al. (2002) possuem exemplos de linfoma difuso de grandes células-B (do inglês, *Diffuse large B-cell lymphoma*, DLBCL). O primeiro conjunto de dados, o DLBCLA, é composto por 141 objetos de DLBCL com 661 atributos cada que podem ser classificadas de acordo com três mecanismos moleculares relevantes: a fosforilação oxidativa (OxPhos) com 49 objetos, respostas das células-B (BCR) com 50 objetos e resposta de hospedeiros (HR), com 42 objetos. O conjunto de dados denominado DLBCLB (Rosenwald et al., 2002) é uma versão pré-processada de 180 objetos de DLBCL, que apresenta 661 atributos (Hoshida et al., 2007). Assim como a base de dados DLBCLA, os objetos desse conjunto são classificados de acordo com os mecanismos moleculares BCR, OxPhos e HR, com, respectivamente, 12, 51 e 87 exemplos. Por fim, o conjunto de dados DLBCLC (Shipp et al., 2002) é composto por 58 objetos de DLBCL, com 3795 atributos cada. A versão desse conjunto de dados usada nesta Tese corresponde a uma versão pré-processada por Hoshida et al. (2007). Essa versão contém duas classes que correspondem a DLBCL já curados (32 objetos) e tumores fatais ou com remissão (26 objetos).

Iris é uma base clássica de análise discriminante de tipos de flores proposta por Fisher (1936). Ela contém 150 exemplos de três espécies diferentes de flores Iris: Iris setosa, Iris virgínica e Iris versicolor. Na Iris, cada espécie de flor tem 50 exemplos, cada um com 4 atributos correspondendo à largura e espessura da pétala e da sépala.

A Tabela 6.4 resume as principais características de cada conjunto de dados utilizados nos testes computacionais desta Tese. Nessa tabela, a primeira e a segunda colunas indicam, respectivamente, o conjunto de dados e o número de objetos do respectivo con-

Tabela 6.4: Tabela com as principais características dos conjuntos de dados.

Conjunto de Dados	♯Objetos	♯Estruturas (♯grupos)	♯Atributos
Golub	72	1 (3)	3571
Leukemia	327	1 (7)	271
Breast	699	1 (2)	9
Novartis	103	1 (4)	1000
Lung	197	1 (4)	1000
MiRNA	218	6 (2,2,3,4,9,20)	217
Proteínas	698	2 (4, 27)	125
Yeast	1484	1 (10)	8
Ecoli	336	1 (8)	7
Iris	150	1 (3)	4
MultiA	104	1 (4)	5565
MultiB	32	1 (4)	5565
BreastA	98	1 (3)	1213
BreastB	48	1 (2)	1213
DLBCLA	141	1 (3)	661
DLBCLB	180	1 (3)	661
DLBCLC	58	1 (2)	3795

junto de dados. A terceira coluna, ♯Estruturas (♯grupos), possui o número de estruturas em cada conjunto de dados e, em parênteses, o número de *clusters* de cada estrutura. A quarta coluna, ♯Atributos, apresenta o número de atributos de cada objeto do conjunto de dados.

6.3.2 Construção do Grafo

Dado um conjunto de dados, como os anteriormente descritos, a construção de um grafo a partir desse conjunto requer algumas considerações. A primeira corresponde à definição de sua estrutura: o que seriam os nós do grafo e o que determinaria as arestas desse grafo e seus pesos associados. A forma mais direta e intuitiva de representar o conjunto de dados em um grafo é considerar que seus nós representam os objetos do conjunto de dados, conectados entre si por arestas e cujos pesos representam uma medida de similaridade entre os objetos (nós). Tendo em vista essa estrutura de grafo, o próximo passo é definir a medida de similaridade a ser utilizada para a definição dos pesos das arestas. Não foram observadas na literatura muitas medidas para avaliação da similaridade entre pares de objetos. Como os objetos dos conjuntos de dados apresentados anteriormente possuem diversos atributos, é necessário definir uma medida que os considere de forma a utilizar toda a informação fornecida pelos dados. Um exemplo de medida de similaridade é dado pela função Gaussiana. Essa função é bastante utilizada para construir grafos para agrupamento de dados. Dado um par de objetos i e j do conjunto de dados, sua

similaridade pode ser calculada por:

$$s_{ij} = e^{-\frac{d(i,j)}{2\sigma^2}}, \tag{6.5}$$

em que σ é um parâmetro ajustado segundo a aplicação e normalização do conjunto de dados e $d(i,j)$ é uma medida de distância entre os objetos i e j. Alguns exemplos de medidas de dissimilaridade que poderiam ser usadas para essa definição são: a distância Euclidiana, a distância Manhattan, a distância Chebyshev e a distância de Mahalanobis. Nesta Tese, a distância Euclidiana é adotada como medida padrão de dissimilaridade entre objetos. O primeiro motivo de sua escolha é o fato da distância Euclidiana ser uma métrica. Uma métrica consiste de uma função não negativa $d(x,y)$ que representa a distância entre dois objetos x e y do espaço métrico correspondente (Gray, 1997). Essa função deve respeitar as seguintes propriedades:

(i) $d(x,y) \leq d(x,z) + d(z,y)$, conhecida como desigualdade triangular;

(ii) $d(x,y) = d(y,x)$, conhecida como propriedade simétrica;

(iii) $d(x,x) = 0$.

Além do mais, a distância Euclidiana é a distância mais utilizada na literatura para esse tipo de problema, e seu valor é representável no plano cartesiano de forma intuitiva. Para calcular a distância Euclidiana entre os objetos de um conjunto de dados, considere a_{ik} o k-ésimo atributo do objeto (ou nó) i e n_a o número de atributos de cada objeto. O cálculo da distância é dado a seguir:

$$d(i,j) = \sqrt{\sum_{k=1}^{n_a}(a_{ik} - a_{jk})^2}. \tag{6.6}$$

Uma maneira de definir uma medida de similaridade a partir de uma medida de dissimilaridade é utilizando o seguinte cálculo:

$$w_{ij} = 1 - \frac{d_{ij}}{1 + d_{max}}, \tag{6.7}$$

em que $d_{max} = \max_{1 \leq i,j \leq N} d(i,j)$, isso é, d_{max} é a maior distância entre quaisquer dois objetos do conjunto de dados.

A partir da medida de similaridade dada pela função Gaussiana ou pela baseada na Equação 6.7, pode-se construir um grafo completo calculando-se a similaridade entre todos

os pares de objetos do conjunto de dados, ou seja, calculando-se a matriz de similaridades. Essa matriz, portanto, irá representar a matriz de pesos das arestas do grafo.

O problema da medida de similaridade baseada na função Gaussiana é a influência que o parâmetro σ exerce na estrutura dos grafos construídos a partir dela. Ng et al. (2002) apresentaram uma maneira de ajustar esse parâmetro, que é por meio de um estudo empírico baseado na consideração de diversos valores para σ, de forma a usar a partição que apresente a menor distorção nos dados. Entretanto, não foi encontrada na literatura nenhuma maneira eficiente de obter valores para o parâmetro σ. Além disso, estudos reportam que a construção do grafo pode causar grandes diferenças nos resultados do algoritmo de agrupamento (Maier e Hein, 2009), podendo conduzir a resultados tendenciosos. O impacto desse parâmetro nas partições obtidas pelos algoritmos propostos nesta Tese é avaliado na Seção 6.3.3.

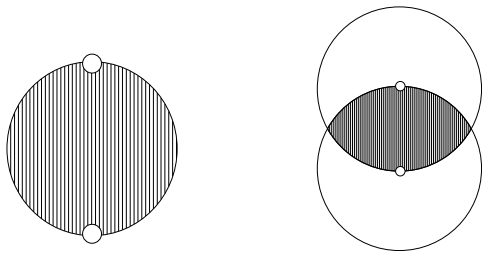
Muitas vezes, o grafo completo não é a melhor forma de se representar os dados em um grafo, pois, devido à grande quantidade de arestas resultantes dessa construção, o algoritmo de agrupamento pode apresentar um alto custo computacional. Por essa razão, existe alternativas para se construir um grafo que não seja completo a partir de um conjunto de dados. Algumas delas são apresentadas a seguir.

Uma construção simples do grafo de dados é definir suas arestas levando-se em consideração um limitante superior para a distância dos objetos, ou seja, um par de nós é adjacente se a distância entre eles é menor ou igual a uma distância d pré-definida. O grafo resultante pode ou não apresentar pesos em suas arestas, definidos por alguma medida de similaridade a partir das distâncias entre os pares de nós já conhecidas. Essa estrutura é conhecida como grafo dos ϵ vizinhos mais próximos (Von Luxburg, 2007).

Outra forma de se construir um grafo é considerar um determinado número κ de vizinhos mais próximos. Nesse caso, a aresta entre dois nós i e j do grafo é considerada apenas se a distância entre i e j for uma das κ mais próximas dentre todos os nós. Essa abordagem tem duas variações: se o grafo não for direcionado, que é o caso dos dados aqui considerados, a aresta pode existir se a distância entre os nós i e j for uma das κ menores com relação ao nó i e ao nó j (vizinhanças mutuamente mais próximas), ou apenas com relação ao objeto i ou apenas com relação ao objeto j (vizinhanças mais próximas). O valor de κ e o cálculo das distâncias devem ser definidos previamente (Jarvis e Patrick, 1973).

Outra abordagem para construção do grafo é dada pela avaliação das medidas de vizinhanças que considera as arestas entre dois nós, como, por exemplo, o grafo Gabriel (GG) e o grafo de vizinhanças relativas, (RNG, do inglês *relative neighborhood graph*).

Para a definição do GG e do RNG, seja $B(x,r)$ uma bola de centro x e raio r



(a) (b)

Figura 6.18: Figura das vizinhanças de grafo. (a) Região do Grafo Gabriel. A região que define a aresta entre dois nós é um círculo cujo centro é o ponto médio dos nós e o diâmetro é a distância entre os nós (região hachurada). Se não houver nenhum outro nó nessa região, então existe uma aresta entre esses nós. (b) Região das vizinhanças relativas. Nesse caso, a região que define a aresta é dada pela região hachurada, que é a intersecção de dois círculos indicados na figura.

definida por $B(x,r) = \{y : d(x,y) < r\}$ em que d é alguma medida de distância. Seja o grafo $G = (V,E)$. A vizinhança do GG é definida por regiões circulares em que uma aresta é adicionada ao grafo se e somente se nenhum ponto do conjunto de dados pertence ao círculo que tenha como diâmetro tal aresta. Dessa forma, o GG pode ser definido da seguinte maneira: a aresta $e = (i,j) \in E$ se e somente se $B(\frac{i+j}{2}, \frac{d(i,j)}{2}) \cap V = \emptyset$. Uma ilustração da vizinhança desse grafo é apresentada na Figura 6.18. O grafo RNG é caracterizado pela seguinte construção: uma aresta $e = (i,j) \in E$ se e somente se $B(i,d(i,j)) \cap B(j,d(i,j)) \cap V = \emptyset$. Uma ilustração da vizinhança do grafo RNG é dada na Figura 6.18.

A escolha da melhor alternativa para a construção de um grafo a partir de um conjunto de dados depende de diversos fatores. Nem sempre há a garantia de que a estrutura adotada é a ideal para a representação de dados ou para a aplicação de um algoritmo de agrupamento. Alguns resultados empíricos podem fornecer apenas indícios de quais são as estruturas mais adequadas para um dado algoritmo. As construções aqui apresentadas são as mais usadas em agrupamento de dados. Tendo isso em vista, alguns testes com conjuntos de dados reais foram realizados e seus resultados são reportados a seguir.

6.3.3 Resultados dos Testes com Bases de Dados Reais

Considerando a discussão sobre a construção dos grafos a partir dos conjuntos de dados a serem utilizados nesse experimento, optou-se por examinar a adoção de diferentes medidas de similaridade para a construção de um grafo completo dos conjuntos de dados.

As partições encontradas para cada uma dessas construções foram comparadas com as partições que fornecem a classificação real dos dados. Para que os testes não ficassem muito redundantes, apenas alguns dos resultados, os julgados mais relevantes para as conclusões sobre a influência da construção do grafo nos resultados, são apresentados. Então, procurou-se utilizar uma maneira menos tendenciosa de construção dos grafos a partir dos conjuntos de dados. Essa representação corresponde a um grafo com pesos que não é completo.

Para se avaliar os resultados obtidos pelos algoritmos testados, utilizou-se a medida *CRand*, que compara numericamente as partições encontradas pelos algoritmos e as classes desses dados. Vale observar que o único algoritmo que necessita da definição do número de *clusters* em sua formulação é o GRASPI. Para reportar os resultados desse algoritmo, foi adotada uma metodologia de geração de partições com um número de *clusters* no conjunto $\{2,\ldots,30\}$. A partição que melhor se aproximar da classificação real dos dados, ou seja, que obtiver o maior *CRand*, é escolhida. Portanto, o que é reportado é o maior *CRand* dentre as partições obtidas pelo GRASPI considerando $k \in \{2,\ldots,30\}$. A Tabela 6.5 apresenta os resultados obtidos usando a construção de um grafo completo com a medida de similaridade dada pela Equação 6.7.

Com relação aos resultados obtidos pelo GRASPI, pode ser observado que os índices *CRand* atingidos pelas suas partições apresentam valores bastante superiores em relação aos outros algoritmos. Ao contrário do que aconteceu nos grafos dos testes da Seção 6.2.2, as partições encontradas nesses grafos completos não apresentaram nós isolados. Pelo que foi observado, os resultados ficam mais confiáveis para essa formulação quando o grafo é completo, assim como acontece com o CCMLI. Ainda na Tabela 6.5, pode ser notado que o GRASPII, que corresponde ao algoritmo para a maximização da modularidade, apresentou índices *CRand* elevados, mas piores do que o GRASPI. O CCMLI também apresentou índices *CRand* altos, sendo competitivo com o GRASPII, ao contrário do que acontecia com as partições encontrados para os grafos simulados de seções anteriores. Isso acontece porque, como já foi discutido na Seção 6.2.2, a formulação baseada no índice *clustering coefficient* de Onnela et al. (2005) apresenta um comportamento mais condizente com a intuição dos grupos quando o grafo a ser estudado é completo. Pelos resultados observados, a formulação baseada no *clustering coefficient* de Barrat et al. (2004) não apresentou índices *CRand* elevados, exceto para a base Iris.

Quando um grafo completo dos conjuntos de dados foi construído usando a função Gaussiana dada pela Equação 6.5 com $\sigma \geq 1$, observou-se partições semelhantes às encontradas no particionamento dos grafos construídos a partir da medida de similaridade calculada pela Equação 6.7. O comportamento da função Gaussiana conforme aumenta o valor de σ é de uma queda mais suave em sua curva, ou seja, a curva fica mais achatada.

Tabela 6.5: Tabela com os valores dos índices *CRand* das partições obtidas por todos os algoritmos propostos com bases reais e utilizando a construção do grafo baseada na similaridade dada pela Equação 6.7. Os valores dos índices *CRand* mais altos estão destacados em negrito.

	GRASPI		GRASPII		CCMLI		CCMLII	
Dados	k	*CRand*	k	*CRand*	k	*CRand*	k	*CRand*
Golub	3	**0.442**	3	0.428	3	0.395	3	0.039
Leukemia	5	0.617	5	0.647	9	**0.689**	3	0.315
Breast	2	0.877	2	**0.882**	2	0.583	1	0
Novartis	4	0.921	4	**0.946**	4	**0.946**	3	0.266
Lung	7	0.170	3	0.164	6	**0.303**	2	0.081
MiRNA	2	**0.510**	2	0.471	8	0.168	2	0.309
MiRNA	5	0.096	2	0.094	8	0.060	2	**0.127**
MiRNA	4	**0.190**	2	0.141	8	0.151	2	0.085
MiRNA	2	**0.503**	2	0.456	8	0.268	2	0.280
MiRNA	4	**0.060**	2	0.037	8	0.048	2	-0.010
MiRNA	18	**0.342**	2	0.097	8	0.315	2	0.090
Proteinas	4	**0.327**	3	0.253	3	0.167	4	0.063
Proteinas	11	**0.167**	3	0.056	3	0.027	4	0.018
Yeast	9	0.150	3	0.090	3	**0.156**	3	0.003
Ecoli	3	0.603	3	**0.664**	8	0.404	2	0.346
Iris	3	**0.756**	2	0.522	4	0.637	3	0.660
MultiA	4	**0.873**	3	0.664	5	0.701	2	0.319
MultiB	7	**0.291**	2	-0.026	2	-0.026	2	0.006
BreastA	2	0.681	2	**0.694**	2	0.653	2	0.409
BreastB	2	**0.321**	2	0.163	3	0.202	2	0.289
DLBCLA	4	**0.407**	3	0.268	4	0.259	2	0.160
DLBCLB	4	0.480	2	0.411	2	**0.565**	2	0.454
DLBCLC	5	**0.314**	2	-0.017	2	-0.017	2	0.002

Dessa maneira, a diferença das distâncias entre objetos é menos acentuada se esses estão próximos ou afastados quando se considera uma curva Gaussiana com um σ alto. Quando o valor do parâmetro σ da função Gaussiana dada pela Equação 6.5 for baixo, a proximidade dos objetos faz com que a sua similaridade seja muito mais alta, enquanto que se os objetos forem distantes, essa distância é traduzida por uma similaridade bem inferior do que no primeiro caso. Em outras palavras, quando o valor de σ for pequeno, nós mais próximos ficam mais próximos, enquanto que nós distantes ficam mais distantes. Tendo isso em vista, foram testados outros valores para esse parâmetro σ. Como exemplo, os resultados da medida *CRand* para $\sigma = 0.03$ são reportados na Tabela 6.6. Nesse caso, devido à diferença de dimensionalidade dos dados, foi necessária uma normalização prévia da matriz de distâncias para que essa apresentasse valores no intervalo [0,1]. Para essa normalização, todos os elementos da matriz de distâncias foram divididos pela sua maior distância antes da normalização.

Tabela 6.6: Tabela com os valores dos índices *CRand* das partições obtidas por todos os algoritmos propostos com bases reais e usando a medida de similaridade dada pela Equacão 6.5 com $\sigma = 0.03$. Os valores dos índices *CRand* mais altos estão destacados em negrito.

	GRASPI		GRASPII		CCMLI		CCMLII	
Dados	k	*CRand*	k	*CRand*	k	*CRand*	k	*CRand*
Golub	5	0.586	10	0.241	9	0.245	2	**0.695**
Leukemia	8	**0.704**	41	0.299	10	0.458	4	0.584
Breast	3	0.747	38	0.090	10	0.175	2	**0.818**
Novartis	9	0.729	19	0.370	10	0.409	4	**0.847**
Lung	5	**0.520**	34	0.043	10	0.120	2	0.135
MiRNA	3	0.450	34	0.049	10	0.157	2	**0.523**
MiRNA	2	**0.106**	34	0.013	10	0.040	2	0.073
MiRNA	6	**0.155**	34	0.029	10	0.115	2	0.128
MiRNA	3	0.506	34	0.061	10	0.217	2	**0.532**
MiRNA	8	**0.109**	34	0.013	10	0.047	2	0.071
MiRNA	14	**0.306**	34	0.293	10	0.244	2	0.107
Proteinas	13	**0.274**	76	0.031	10	0.111	2	0.193
Proteinas	14	0.071	76	0.046	10	**0.077**	2	0.038
Yeast	10	0.109	60	0.052	9	0.146	4	**0.168**
Ecoli	13	**0.722**	34	0.095	10	0.283	2	0.454
Iris	3	0.680	11	0.382	10	0.340	3	**0.746**
MultiA	13	**0.599**	17	0.361	7	0.568	3	0.563
MultiB	6	0.213	5	**0.274**	8	0.213	2	0.084
BreastA	10	**0.495**	3	0.431	9	0.248	3	0.431
BreastB	7	0.416	3	0.445	9	0.138	2	**0.500**
DLBCLA	15	**0.296**	15	0.155	10	0.088	2	0.227
DLBCLB	6	0.380	24	0.141	9	0.189	2	**0.547**
DLBCLC	2	**0.008**	5	0.002	10	-0.008	3	-0.016

Na Tabela 6.6 nota-se um padrão de comportamento diferente do até agora observado nos valores dos índices *CRand* das partições obtidas pelos algoritmos propostos nesta Tese. O GRASPI obteve índices altos na maioria das partições, apresentando mais vezes o maior valor do índice *CRand*, 12 vezes. Isso ocorreu provavelmente porque os grafos particionados são completos. O GRASPII apresentou um comportamento regular, não tendo em nenhuma base um índice *CRand* superior a 0.5. O CCMLI apresentou resultados ligeiramente melhores do que os do GRASPII e piores do que os do GRASPI e CCMLII. Os valores dos índices *CRand* das partições encontradas pelo CCMLII foram muito bons. Para a maioria dos casos, esses valores foram bem altos, sendo que 9 deles foram maiores do que os dos outros algoritmos.

Na tentativa de evitar o problema de encontrar resultados tendenciosos por causa da construção indevida dos grafos a partir dos dados, propôs-se a construção do grafo dos κ vizinhos mais próximos (a abordagem de vizinhanças mais próximas), e posterior

observação gráfica dos grafos produzidos a partir dessa abordagem. Para a observação gráfica, os grafos resultantes foram plotados com os rótulos de suas classes em seus nós. Foi analisado se os grafos representados por essa abordagem apresentavam conectividade em suas classes, ou seja, tinham características de agrupamento em sua classificação. Como resultado, a maioria dos grafos construídos por meio dessa abordagem apresentaram uma estrutura bem definida de agrupamento, como pode ser verificado no Apêndice A. O valor adotado para κ foi 15, obtido por meio de testes preliminares que avaliaram visualmente as características dos grafos obtidos.

Tabela 6.7: Tabela com os valores dos índices *CRand* das partições obtidas por todos os algoritmos propostos com bases reais e utilizando o grafo dos 15 vizinhos mais próximos e com pesos. Os valores dos índices *CRand* mais altos estão destacados em negrito.

	GRASPI		GRASPII		CCMLI		CCMLII	
Dados	k	*CRand*	k	*CRand*	k	*CRand*	k	*CRand*
Golub	11	0.079	4	0.424	10	0.154	2	**0.592**
Leukemia	11	0.521	6	0.786	10	0.292	6	**0.792**
Breast	2	-0.001	47	0.093	4	0.027	8	**0.164**
Novartis	6	0.296	4	0.946	9	0.552	4	**0.973**
Lung	12	**0.430**	6	0.304	9	0.036	2	0.188
MiRNA	12	0.156	6	0.153	10	0.102	8	**0.191**
MiRNA	13	0.004	6	**0.095**	10	0.029	8	0.067
MiRNA	13	0.089	6	**0.149**	10	0.041	8	0.124
MiRNA	12	0.218	6	**0.241**	10	0.088	8	0.224
MiRNA	11	**0.073**	6	-0.001	10	-0.004	8	0.032
MiRNA	12	0.016	6	0.282	10	0.120	8	**0.333**
Proteinas	4	0.001	10	0.144	5	0.009	6	**0.253**
Proteinas	4	0.000	10	**0.083**	5	0.008	6	0.053
Yeast	5	0.000	25	**0.087**	4	0.000	2	0.069
Ecoli	7	-0.001	7	0.394	10	0.138	3	**0.709**
Iris	14	0.393	5	0.510	10	0.177	3	**0.732**
MultiA	11	0.262	4	0.767	10	0.283	4	**0.773**
MultiB	11	**0.010**	2	-0.025	3	-0.029	2	-0.027
BreastA	11	0.332	4	0.334	10	0.091	2	**0.446**
BreastB	11	0.034	2	**0.500**	9	0.155	2	**0.500**
DLBCLA	15	0.007	4	**0.369**	10	0.035	2	0.209
DLBCLB	13	0.019	5	0.303	10	0.079	2	**0.605**
DLBCLC	2	-0.006	3	-0.020	10	-0.013	1	**0.000**

Considerando essa construção de grafo ponderado, pode ser observado que ambos os algoritmos GRASPI e CCMLI apresentaram resultados inferiores aos obtidos pelos algoritmos GRASPII e CCMLII. O comportamento indesejado desses algoritmos em grafos não completos e com pesos se repetiu e, portanto, pode ser uma tendência dessas formulações. Portanto, caso deseja-se utilizá-las por apresentarem bons resultados quando o grafo é completo, uma alternativa é pré-processar o grafo para torná-lo completo, usando,

por exemplo, a abordagem dos caminhos mais próximos discutida anteriormente.

Com relação aos resultados obtidos pelos algoritmos GRASPII e CCMLII, foi observado que ambos apresentaram índices *CRand* altos e baixos nos mesmos casos. Entretanto, considerando os índices mais altos, foi visto que o algoritmo CCMLII apresentou classificações mais compatíveis com as dos dados estudados. Alguns exemplos são as bases DLBCLB e Ecoli, nas quais o algoritmo CCMLII apresentou partições com índices *CRand* de 0.605 e 0.709, enquanto que para o GRASPII foram de 0.303 e 0.394, respectivamente.

Como esses problemas são de *classificação de dados*, não é esperado que os resultados atingidos coincidam com o agrupamento dos dados. Ainda mais porque os dados podem apresentar mais de uma classificação, como é o caso do conjunto de dados MiRNA. Entretanto, é observado que o problema de classificação de dados, que é um problema de reconhecimento de padrões, apresenta certas semelhanças com o problema de agrupamento de dados, pois em dados de classificação podem ser observados agrupamentos e, muitas vezes, os grupos encontrados por meio de algoritmo de agrupamento têm grande aproximação com as classes dos dados.

6.4 Considerações Finais

Foi possível concluir por meio dos testes realizados nesta Tese que os algoritmos GRASPI e CCMLI apresentaram um comportamento semelhante no sentido de ramificar muito os *clusters* quando o grafo a ser particionado não é completo. Por meio de alguns estudos realizados, propôs-se uma alternativa para amenizar a limitação do CCMLI nesses casos. Apesar do aumento do custo computacional do problema, considerar a matriz de similaridades baseada na construção dos caminhos mais próximos para todos pares de nós do grafo pode ser uma alternativa para o GRASPI. Como essa matriz produz uma relação numérica para todos pares de nós do grafo, o resultado das partições calculadas a partir dela pode ser melhor para o problema de agrupamento por agregar mais informação global do grafo.

Com relação aos algoritmos GRASPII e CCMLII, pode-se concluir que quando o grafo a ser particionado é ponderado, mas não completo, ambos apresentam resultados muito bons, sendo que o uso de um algoritmo de busca local mais intensivo no CCMLII levou a um melhor comportamento do CCMLII com índices de acertos mais altos do que o GRASPII. Entretanto, o GRASPII obteve altos índices *CRand* para grafos completos usando a medida de similaridade dada pela Equação 6.7, comportamento não observado pelo CCMLII. Como no problema de detecção de comunidades os grafos a serem manipulados dificilmente são completos, esse comportamento do CCMLII não é necessariamente

um problema. Se for necessário manipular um grafo completo, é possível pré-processar seus dados de forma a utilizar uma medida que torne as distâncias mais sensíveis, como a função Gaussiana com valores de σ baixo, ou eliminando arestas por meio da estratégia de vizinhos mais próximos. Uma alternativa é adotar o algoritmo CCMLI quando o grafo for completo e ou o CCMLII, caso contrário.

Capítulo 7

Conclusões e Perspectivas Futuras

Este capítulo relembra os objetivos e destaca os principais resultados desta Tese. Ao final, são fornecidas sugestões para o direcionamento de pesquisas futuras relacionadas à continuidade deste trabalho.

7.1 Considerações Finais

Esta Tese teve como objetivo principal apresentar contribuições na área de aprendizado não-supervisionado por meio da proposição de métodos e modelos matemáticos para o problema de agrupamento de dados em grafos. Em uma primeira etapa, foram analisados modelos de particionamento de grafos já publicados na literatura e dois deles foram selecionados para serem estudados nesta Tese. Para os modelos selecionados, ou seja, o de maximização da soma das arestas intra-*cluster* e o de maximização da modularidade das partições, foram propostas metaheurísticas GRASP. Em uma primeira etapa de experimentos, os resultados das metaheurísticas foram comparados com os de heurísticas da literatura comumente utilizadas para resolver esses problemas. Os resultados comparativos apontaram que as soluções das metaheurísticas propostas apresentaram uma melhor qualidade do que as da literatura, com resultados bem superiores. Com relação ao desempenho das metaheurísticas, a melhoria na qualidade das soluções foi obtida em um tempo computacional superior ao de heurísticas encontradas na literatura. Na segunda etapa de experimentos, as partições encontradas foram comparadas com a classificação

do conjunto de grafos modulares que apresentavam uma clara estrutura de agrupamento. Foi observado que as partições encontradas para o modelo de maximização da modularidade das partições foram bem satisfatórias, apresentando índices de correlação bem altos com a classificação dos grafos. Com relação às partições encontradas para o modelo de maximização da similaridade intra-*cluster*, uma limitação do modelo, já apontada na literatura, foi confirmada: a de soluções com *clusters* com único nó. Entretanto, observou-se que quando o grafo particionado por esse modelo era completo, a limitação do modelo anteriormente mencionada não era observada. Pelo contrário, as soluções encontradas pela metaheurística tinham uma boa qualidade e quando a metaheurística foi aplicada a dados de classificação, os resultados foram bem consistentes (Nascimento et al., 2010b). O que se pode concluir por meio dessas observações é que para esse último modelo, um pré-processamento dos dados de forma a se ter uma matriz de similaridades indicando uma relação para todos os pares de nós do grafo pode ser uma boa alternativa para o seu uso. Uma sugestão foi apresentada para esse pré-processamento, que consiste na utilização da matriz das distâncias mais próximas entre os nós do grafo original adaptada para matriz de similaridades.

A segunda contribuição desta Tese foi a proposta de modelos de agrupamento de dados em grafos que têm a capacidade de detectar partições sem a necessidade de previamente fornecer o número de *clusters*, baseando-se na conectividade dos nós intra-grupos no grafo. Para atingir tal objetivo, uma medida originalmente proposta para avaliar a tendência de agrupamento de nós de um grafo, conhecida como *clustering coefficient* (Watts e Strogatz, 1998), foi usada como ponto de partida da nova formulação matemática. Essa medida foi inicialmente introduzida para grafos sem pesos. Anos depois de ter sido proposta, duas generalizações para grafos ponderados foram introduzidas, uma proposta por Barrat et al. (2004) e a outra por Onnela et al. (2005). Nesta Tese, ambas as generalizações dessa medida foram estudadas e implementadas para encontrar *partições* de um grafo (por meio de um modelo matemático que as maximize). De acordo com os experimentos computacionais realizados com centenas de grafos, a formulação matemática proposta que maximiza a medida introduzida por Barrat et al. (2004) apresentou resultados bem satisfatórios e condizentes com a conhecida classificação dos dados. Inclusive, essa formulação forneceu, por meio de uma heurística multi-nível, partições mais consistentes do que as partições encontradas pela formulação que maximiza a modularidade das partições. O destaque deste trabalho é que essa última formulação é uma das mais usadas na área de detecção de comunidades, havendo centenas publicações em revistas de alto impacto sobre o assunto (Clauset et al., 2004; Newman, 2006). A outra formulação, baseada na medida de Onnela et al. (2005) foi mais bem sucedida quando o grafo particionado era completo, assim como acontece com o modelo de maximização da soma das arestas intra-*cluster*.

As principais contribuições desta Tese são sumarizadas a seguir.

- A proposta de uma metaheurística GRASP para o modelo de maximização da similaridade intra-*cluster*, que apresentou um bom desempenho com relação ao tempo de processamento e à qualidade das soluções encontradas. Esse trabalho foi publicado e sua referência é Nascimento et al. (2010b).
- A proposta de uma metaheurística GRASP, agora para o modelo de maximização da modularidade das partições. Esse trabalho também obteve bons resultados, no entanto é computacionalmente custoso, principalmente em problemas nos quais o grafo possui mais de 500 nós. Para esses problemas, encontrar a melhor solução da metaheurística pode levar mais de 20 minutos, dependendo do tamanho do grafo. Esse trabalho foi desenvolvido em estágio na Grécia, com a colaboração do Prof Dr Leonidas Pitsoulis, e resultou em um artigo que está em fase de preparação e que será submetido a um periódico internacional.
- A proposta de uma formulação matemática baseada nas medidas *clustering coefficient* para grafos com e sem pesos. Parte desse trabalho resultou em um artigo que foi submetido a um periódico internacional. A proposta do modelo de maximização do *clustering coefficient* baseado na medida introduzida por Barrat et al. (2004) foi recentemente finalizada. Um artigo reportando os seus resultados está sendo elaborado para ser submetido a um periódico internacional.

Foram desenvolvidos trabalhos adicionais que não utilizam metaheurísticas e, portanto, são parcialmente relacionados com o tema do doutorado, não descritos nesta Tese para deixá-la mais coesa. Esses trabalhos são resumidos nos itens a seguir.

- A preparação de um *survey* sobre algoritmos de agrupamento baseados em Teoria Espectral em Grafos. Nesse trabalho, foram analisados algoritmos baseados em relaxação espectral de problemas de particionamento de grafos. Testes computacionais com esses algoritmos foram realizados para verificar o seu comportamento. O trabalho resultante dessa pesquisa foi submetido a um periódico internacional.
- Ainda relacionado ao tema de Teoria Espectral em Grafos, um algoritmo *ensemble*, que combina partições de agrupamento, foi desenvolvido e publicado em anais de congresso (Nascimento et al., 2009).
- O estudo e desenvolvimento de uma heurística Lagrangiana para o problema de minimização das distâncias entre os objetos e seus medóides, conhecido como o problema de minimização dos k-medóides. Esse artigo foi submetido a um periódico nacional.

7.2 Pesquisas Futuras

Como pesquisas futuras, pretende-se melhorar o desempenho das metaheurísticas propostas, tentando a sua adaptação para algoritmos multi-níveis. Além disso, será desenvolvida a sua hibridização com algoritmos espectrais, que indicaram ter um custo computacional baixo.

Como um objetivo mais teórico, almeja-se desenvolver um estudo formal da formulação proposta (a de maximização da medida baseada no *clustering coefficient*) e do modelo de maximização da modularidade das partições. A medida de modularidade das partições já vem sendo estudada e alguns de seus aspectos negativos já foram reportados (Porter et al., 2009). Deseja-se avaliar as formulações matemáticas propostas nesta Tese, que são de maximização do *clustering coefficient*, de forma a tornar mais clara a natureza das partições e dos tipos de grupos por elas encontradas. Se possível, procurar-se-á desenvolver um estudo sobre o modelo não-linear de forma a encontrar soluções ótimas para problemas pequenos. Além disso, deseja-se definir suas limitações e vantagens, propondo alternativas para contornar suas limitações, se existentes.

Apêndice

A

Grafos dos dados

A seguir, será apresentada uma visualização gráfica dos grafos dos k vizinhos mais próximos por meio da ferramenta *Graphopt* do pacote *igraph* do R-*project*. Com as bases de dados descritas anteriormente, será observado que o grafo dos k vizinhos mais próximos pode ser bastante informativo para agrupamento em grafos.

O valor de k levado em consideração é 15, ajustado por meio de testes preliminares, e os grafos estão apresentados para cada uma das classificações. Os rótulos das classificações reais do conjunto de dados estão denotados por cores diferentes.

Pode-se observar que o grafo dos k vizinhos mais próximos apresentou uma boa representação para a maioria dos dados. Entretanto, em algumas bases como, por exemplo, a Proteínas e a Yeast, é difícil de identificar as suas partições por meio desses grafos.

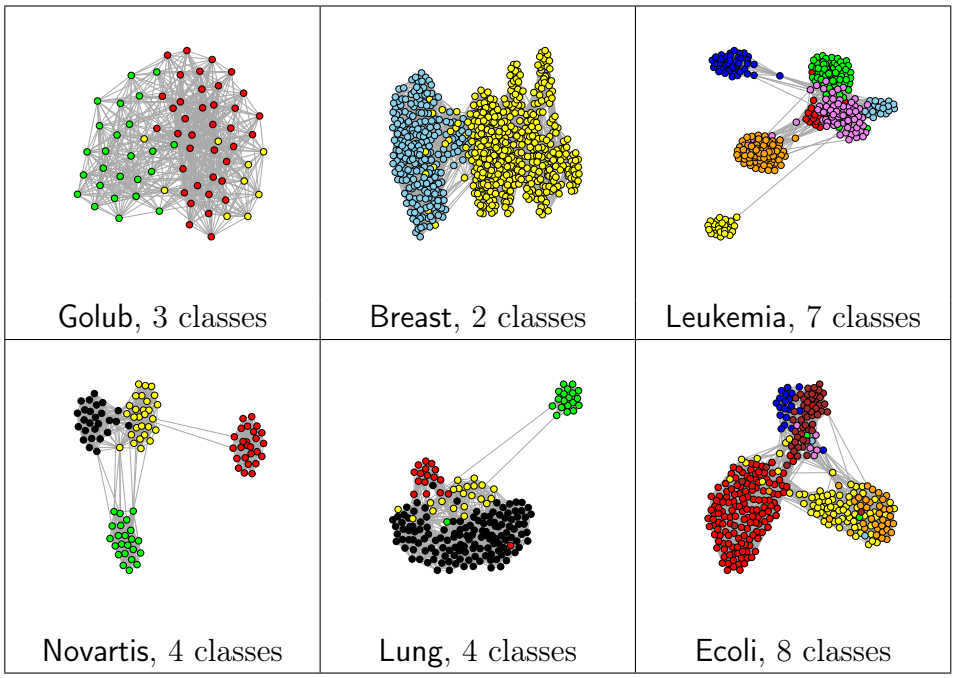


Figura A.1: Grafos dos 15 vizinhos mais próximos de algumas bases de dados biológicas.

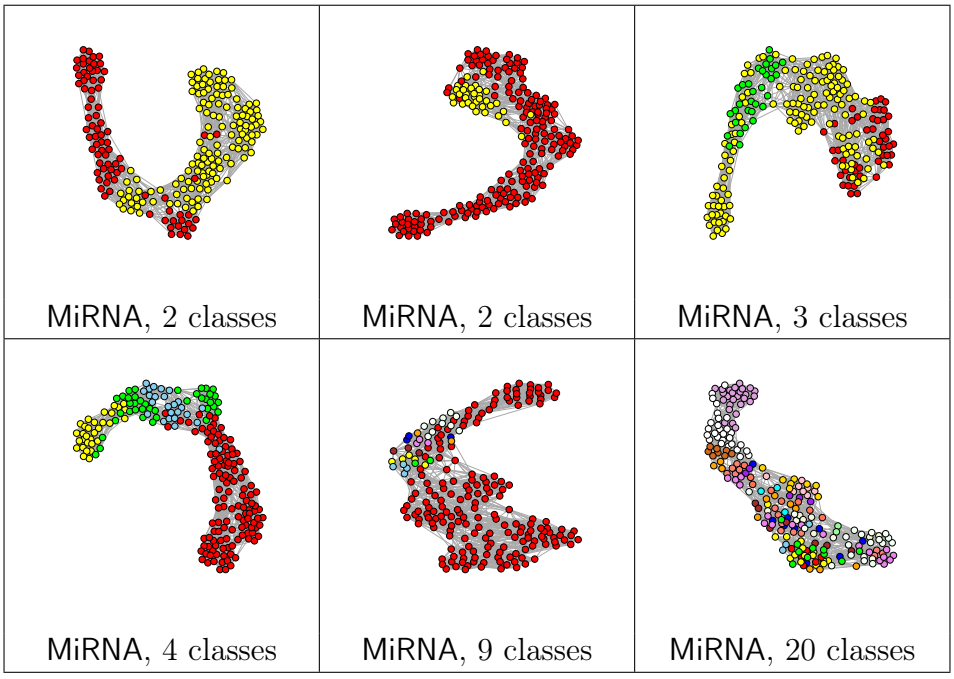


Figura A.2: Grafos dos 15 vizinhos mais próximos da base de dados MiRNA.

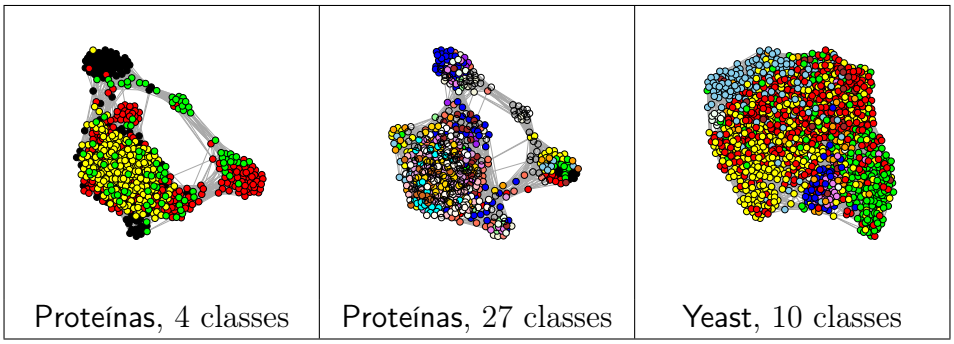


Figura A.3: Grafos dos 15 vizinhos mais próximos das bases de Proteínas e Yeast.

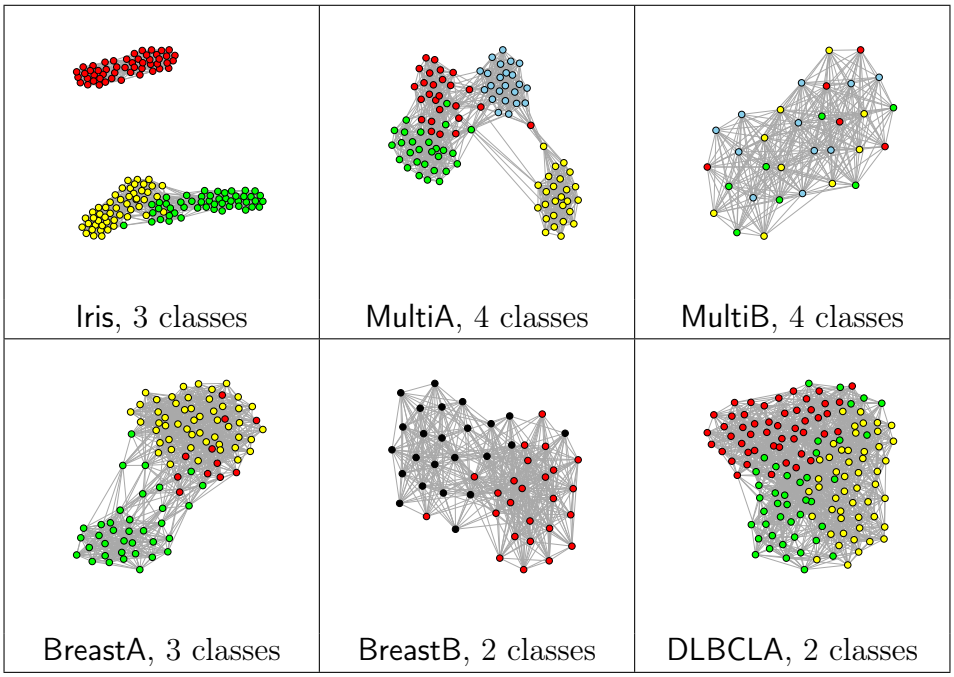


Figura A.4: Grafos dos 15 vizinhos mais próximos de algumas bases de dados.

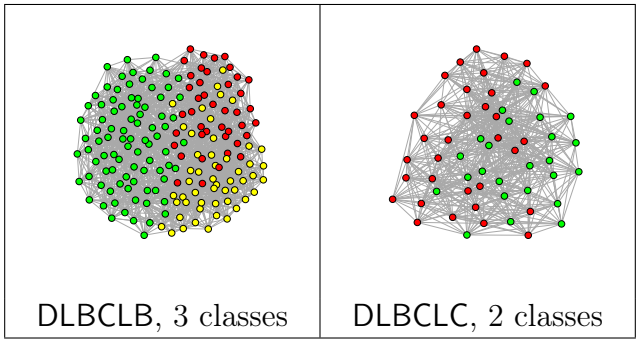


Figura A.5: Grafos dos 15 vizinhos mais próximos de duas bases de dados.

Referências Bibliográficas

Agarwal, G.; Kempe, D. Modularity-maximizing graph communities via mathematical programming. *European Physical Journal B*, v. 66, p. 409–418, 2008.

Alpert, C.; Kahng, A. B.; Yao, S. Z. Spectral partitioning with multiple eigenvectors. *Discrete Applied Mathematics*, v. 90, p. 3–26, 1999.

Alpert, C. J.; Kahng, A. B. Multi-way partitioning via spacefilling curv es and dynamic programming. In: *31st ACM/IEEE Design Automation Conference*, 1994.

Alpert, C. J.; Yao, S.-Z. *Spectral partitioning: The more eigenvectors, the better.* Relatório Técnico 940036, UCLA CS Department Technical Report, 1994.

Areibi, S.; Vannelli, A. A GRASP clustering technique for circuit partitioning. In: Gu, J.; Pardalos, P., eds. *Satisfiability problems*, v. 35 de *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, p. 711–724, 1997.

Barnard, S. T.; Simon, H. D. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, v. 6, p. 101–107, 1994.

Barrat, A.; Barthelemy, M.; Pastor-Satorras, R.; Vespignani, A. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences*, v. 101, p. 3747–3752, 2004.

Beliakov, G.; King, M. Density based fuzzy c-means clustering of non-convex patterns. *European Journal of Operational Research*, v. 127, p. 717–728, 2006.

Bennett, K. P.; Mangasarian, O. L. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, v. 1, p. 23–34, 1992.

Bhattacharjee, A.; Richards, W. G.; Staunton, J.; Li, C.; Monti, S.; Vasa, P.; Ladd, C.; Beheshti, J.; Bueno, R.; Gillette, M.; Loda, M.; Weber, G.; Mark, E. J.; Lander, E. S.; Wong, W.; Johnson, B. E.; Golub, T. R.; Sugarbaker, D. J.; Meyerson, M. Classification of human lung carcinomas by

mrna expression profiling reveals distinct adenocarcinoma sub-classes. *Proc Natl Acad Sci USA*, v. 98, n. 24, p. 13790–5, 2001.

Boccaletti, S.; Latora, V.; Moreno, Y.; Chavez, M.; Hwang, D.-U. Complex networks: Structure and dynamics. *Physics Reports*, v. 424, p. 175–308, 2006.

Boginski, V.; Butenko, S.; Pardalos, P. M. Mining market data: A network approach. *Computers & Operations Research*, v. 33, p. 3171–3184, 2006.

Brandes, U.; Delling, D.; Gaertler, M.; Görke, R.; Hoefer, M.; Nikolosk, Z.; Wagner, D. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, v. 20, p. 172–188, 2008.

Cano, J.; Cordón, O.; Herrera, F.; Sánchez, L. *A GRASP algorithm for clustering.* Lecture Notes in Computer Science Springer, p. 214–223, 2002.

Chan, P. K.; Schlag, M. D. F.; Zien, J. Y. Spectral k-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 13, p. 1088–1096, 1994.

Chung, F. R. K. *Spectral graph theory.* N. 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1994.

Clauset, A.; Newman, M. E. J.; Moore, C. Finding community structure in very large networks. *Phys. Rev. E*, v. 70, n. 6, p. 066111, 2004.

Cormack, R. M. A review of classification. *Journal of the Royal Statistical Society*, v. 134, p. 321–367, 1971.

Crowston, W. B.; Glover, F.; Thompson, G. L.; Trawick, J. D. *Probabilistic and parametric learning combinations of local job shop scheduling rules.* Carnegie Mellon University, Pittsburgh, PA, 1963.

Cvetković, D.; Doob, M.; Sachs, H. *Spectra of graphs: Theory and application.* Academic Press, New York, 1979.

de Abreu, N. M. M. Teoria espectral dos grafos: um híbrido entre álgebra linear e matemática discreta e combinatória com origens em química quântica. *TEMA Tend. Mat. Apl. Comput.*, v. 6, p. 1–10, 2005.

Dhillon, I. S.; Guan, Y.; Kulis, B. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 29, n. 11, p. 1944–1957, 2007.

Diestel, R. *Graph theory.* Springer-Verlag, 1997.

Ding, C.; He, X.; Meraz, R. F.; Holbrook, S. R. A unified representation of multiprotein complex data for modeling interaction networks. *Proteins: Structure, Function, and Bioinformatics*, v. 57, p. 99–108, 2004.

Ding, C.; He, X.; Zha, H.; Gu, M.; Simon, H. A min-max cut algorithm for graph partitioning and data clustering. In: *Proceedings IEEE International Conference on Data Mining: ICDM 2001*, 2001.

Ding, C. H. Q.; Dubchak, I. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, v. 17, p. 349–358, 2001.

Dolan, E. D.; Moré, J. J. Benchmarking optimization software with performance profiles. *Math. Program., Ser. A*, v. 91, p. 201–213, 2002.

Dudoit, S.; Fridlyand, J.; Speed, T. P. *Comparison of discrimination methods for the classification of tumors using gene expression data.* Relatório Técnico, Department of Statistics, UC Berkeley, 2000.

Famili, A. F.; Liu, G.; Liu, Z. Evaluation and optimization of clustering in gene expression data analysis. *Bioinformatics*, v. 20, p. 1535–1545, 2004.

Feo, T.; Resende, M. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, v. 8, p. 67–71, 1989.

Feo, T.; Resende, M. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133, 1995.

Festa, P.; Resende, M. G. C. GRASP: An annotated bibliography. In: Ribeiro, C. C.; Hansen, P., eds. *Essays and Surveys on Metaheuristics*, Kluwer Academic Publishers, p. 325–367, 2002.

Fiedler, M. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, v. 25, p. 619–633, 1975.

Filippone, M.; Camastra, F.; Masulli, F.; Rovetta, S. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, v. 41, p. 176–190, 2008.

Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, v. 7, p. 179–188, 1936.

Franti, P.; Kivijarvi, J.; Kaukoranta, T.; Nevalainen, O. Genetic algorithms for large scale clustering problems. *The Computer Journal*, v. 40, n. 9, 1997.

Gabow, H. *Implementation of algorithms for maximum matching on nonbipartite graphs.* Tese de Doutoramento, Stanford University, 1973.

Girvan, M.; Newman, M. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, v. 99, p. 7821–7826, 2002.

Glover, F.; Laguna, M. *Tabu search.* Kluwer Academic Publishers, 1997.

Glover, F.; Laguna, M. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, v. 29, n. 3, p. 653–684, 2000.

Goldberg, D. *Genetic algorithms in search.* Addison-Wesley, 1989.

Golub, T. R.; Slonim, D. K.; Tamayo, P.; Huard, C.; Gaasenbeek, M.; Mesirov, J. P.; Coller, H.; Loh, M. L.; Downing, J. R.; Caligiuri, M. A.; Bloomfield, C. D.; Lander, E. S. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, v. 286, n. 5439, p. 531–7, 1999.

Gray, A. *Metrics on surface*, cáp. 15. 2nd ed Modern Differential Geometry of Curves and Surfaces with Mathematica, p. 341–358, 1997.

Hagen, L.; Kahng, A. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design*, v. 11, p. 1074–1088, 1992.

Hall, K. M. An r-dimensional quadratic placement algorithm. *Management Science*, v. 17, p. 219–229, 1970.

Handl, J.; Knowles, J. *Multiobjective clustering with automatic determination of the number of clusters*. Relatório Técnico TR-COMPSYSBIO-2004-02, UMIST, Manchester, UK, 2004.

Hansen, P.; Jaumard, B. Cluster analysis and mathematical programming. *Mathematical Programming*, v. 79, p. 191–215, 1997.

Hansen, P.; Mladenović, N. J-means: a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, v. 34, p. 405–413, 2001.

Hart, J.; Shogan, A. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, v. 6, p. 107–114, 1987.

Hartuv, E.; Shamir, R. A clustering algorithm based on graph connectivity. *Information Processing Letters*, v. 76, p. 175–181, 2000.

Hendrickson, B.; Leland, R. W. A multi-level algorithm for partitioning graphs. In: *Supercomputing*, 1995.

Higham, D. J.; Kalna, G.; Vass, J. K. Spectral analysis of two-signed microarray expression data. *Mathematical Medicine and Biology*, v. 24, p. 131–148, 2007.

Hlaoui, A.; Wang, S. Median graph computation for graph clustering. *Soft Computing - A Fusion of Foundations Methodologies and Applications*, v. 10, p. 47–53, 2006.

Hoshida, Y.; Brunet, J.-P.; Tamayo, P.; Golub, T. R.; Mesiro, J. P. Subclass mapping: Identifying common subtypes in independent disease data sets. *PLOS one*, v. 2, n. 11, p. e1195, 2007.

Hubert, L.; Arabie, P. Comparing partitions. *Journal of Classification*, v. 2, p. 193–218, 1985.

Huttenhower, C.; Flamholz, A. I.; Landis, J. N.; Sahi, S.; Myers, C. L.; Olszewski, K. L.; Hibbs, M. A.; S., N. O.; Troyanskaya, O. G.; Coller, H. A. Nearest neighbor networks: clustering expression data based on gene neighborhoods. *BMC Bioinformatics*, v. 8, p. 250, 2007.

ILOG *Cplex 11.2 reference manual.* ILOG: France, v. 11.2, 2007.

Jain, A. K.; Murty, M. N.; Flynn, P. J. Data clustering: a review. *ACM Comput. Surv.*, v. 31, p. 264–323, 1999.

Jarvis, R. A.; Patrick, E. A. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, v. c-22, p. 1025–1073, 1973.

KARYPIS, G.; KUMAR, V. Parallel multilevel graph partitioning. In: *Proceedings of the International Parallel Processing Symposium*, 1996.

KARYPIS, G.; KUMAR, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, v. 20, n. 1, p. 359–392, 1998.

KAUFMAN, L.; ROUSSEEUW, P. J. *Finding groups in data: an introduction to cluster analysis*. New York: Wiley, 1990.

KAWAJI, H.; TAKENAKA, Y.; MATSUDA, H. Graph-based clustering for finding distant relationships in a large set of protein sequences. *Bioinformatics*, v. 20, n. 2, p. 243–252, 2004.

KERNIGHAN, B. W.; LIN, S. An efficient heuristic procedure for partitioning graphs. *Bell Sys. Tech. J.*, v. 49, p. 291–307, 1970.

KIRCHHOFF, G. Über die auflösung der gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer ströme geführt wird. *Ann. Phys. Chem.*, v. 72, p. 497–508, 1847.

KOCHENBERGER, G.; GLOVER, F.; ALIDAEE, B.; WANG, H. Clustering of microarray data via clique partitioning. *Journal of Combinatorial Optimization*, v. 10, p. 77–92, 2005.

KRAUSE, A.; STOYE, J.; VINGRON, M. Large scale hierarchical clustering of protein sequences. *BMC Bioinformatics*, v. 6, p. 15, 2005.

LEIGHTON, T.; RAO, S. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In: *Annual IEEE Symposium on Foundations of Computer Science*, 1988, p. 422–431.

LORENA, L. A. N.; FURTADO, J. C. Constructive genetic algorithm for clustering problems. *Evolutionary Computation*, v. 9, n. 3, p. 309–328, 2001.

LU, J.; GETZ, G.; MISKA, E. A.; ALVAREZ-SAAVEDRA, E.; LAMB, J.; PECK, D.; SWEET-CORDERO, A.; EBERT, B. L.; MAK, R. H.; FERRANDO, A. A.; DOWNING, J. R.; JACKS, T.; HORVITZ, R. R.; GOLUB, T. R. Microrna expression profiles classify human cancers. *Nature*, v. 435, n. 7043, p. 834–838, 2005.

MAIER, M., U. V. L.; HEIN, M. Influence of graph construction on graph-based clustering measures. In: KOLLER, D., D. S. Y. B. L. B. C. R. H., ed. *Advances in Neural Information Processing Systems, Proceedings of the 2008 Conference*, 2009, p. 1025–1032.

MARINAKIS, Y.; MARINAKI, M.; DOUMPOS, M.; MATSATSINIS, N.; ZOPOUNIDIS, C. A hybrid stochastic genetic-GRASP algorithm for clustering analysis. *Operational Research*, v. 8, p. 22–46, 2008.

MOHAR, B. The laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, v. 2, p. 871–898, 1991.

Monti, S.; Savage, K. J.; Kutok, J. L.; Feuerhake, F.; Kurtin, P.; Mihm, M.; Wu, B.; Pasqualucci, L.; Neuberg, D.; Aguiar, R. C. T.; Dal Cin, P.; Ladd, C.; Pinkus, G. S.; Salles, G.; Harris, N. L.; Dalla-Fave Molecular profiling of diffuse large b-cell lymphoma identifies robust subtypes including one characterized by host inflammatory response. *Blood*, v. 105, p. 1851 – 61, 2005.

Monti, S.; Tamayo, P.; Mesirov, J.; Golub, T. *Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data*. Relatório Técnico, Broad Institute/MIT - Kluwer Academic Publishers, 2003.

Nakai, K.; Kanehisa, M. Expert system for predicting protein localization sites in gram-negative bacteria. *PROTEINS: Structure, Function, and Genetics*, v. 11, p. 95–110, 1991.

Nascimento, M. C. V.; Resende, M. C. G.; Toledo, F. M. B. GRASP with path-relinking for the multi-plant capacitated lot sizing problem. *European Journal of Operational Research*, v. 200, p. 747–754, 2010a.

Nascimento, M. C. V.; Toledo, F. M. B.; Carvalho, A. C. P. L. F. Consensus clustering using spectral theory. In: *Advances in Neuro-Information Processing*, Springer Berling / Heidelberg, 2009, p. 461–468 (*Part I - Lecture Notes of Computer Science*, v.5506).

Nascimento, M. C. V.; Toledo, F. M. B.; Carvalho, A. C. P. L. F. Investigation of a new GRASP-based clustering algorithm applied to biological data. *Computers & Operations Research*, v. 37, p. 1381–1388, 2010b.

Newman, M. E. J. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, v. 74, p. 036–104, 2006.

Ng, A.; Jordan, M.; Weiss, Y. On spectral clustering: analysis and an algorithm. In: Dietterich, T. G.; Becker, S.; Ghahramani, Z., eds. *Advances in Neural Information Processing Systems*, MIT Press Cambridge, MA, 2002, p. 849–856.

Noack, A.; Rotta, R. Multi-level algorithms for modularity clustering. In: *Proceedings of the 8th International Symposium on Experimental Algorithms (SEA 2009)*, v. 5526 de *Lecture Notes in Computer Science*, Springer, p. 257–268, 2009.

Oliveira, C.; Pardalos, P.; Resende, M. GRASP with path-relinking for the quadratic assignment problem. In: Ribeiro, C.; Martins, S., eds. *Efficient and Experimental Algorithms*, v. 3059, Springer-Verlag, p. 356–368, 2004.

Oliveira, S.; Seok, S. Multilevel approaches for large scale proteomic networks. *International Journal of Computer Mathematics*, v. 84, p. 683–695, 2007.

Oliveira, S.; Seok, S. A matrix-based multilevel approach to identify functional protein modules. *Int. J. Bioinformatics Research and Applications*, v. 4, p. 11–27, 2008.

Onnela, J.-P.; Saramäki, J.; Kertész, J.; Kaski, K. Intensity and coherence of motifs in weighted complex networks. *Phys. Rev. E*, v. 71, p. 065103(R), 2005.

Opsahl, P. Clustering in weighted networks. *Social Networks*, v. in press, 2009.

Pal, N. R.; Biswas, J. Cluster validation using graph theoretic concepts. *Pattern Recognition*, v. 30, p. 847–857, 1997.

Papadimitriou, C. H.; Steiglitz, K. *Combinatorial optimization: Algorithms and complexity*. Second ed. Dover Publications, Inc., 1998.

Porter, M. A.; Onnela, J.-P.; Mucha, P. J. Communities in networks. *Notices of the AMS*, v. 56, p. 1082–1097, 2009.

Ramaswamy, S.; Tamayo, P.; Rifkin, R.; Mukherjee, S.; Yeang, C. H.; Angelo, M.; Ladd, C.; Reich, M.; Latulippe, E.; Mesirov, J. P.; Poggio, T.; Gerald, W.; Loda, M.; Lander, E. S.; Golub, T. R. Multiclass cancer diagnosis using tumor gene expression signatures. *Proc Natl Acad Sci U S A*, v. 98, n. 26, p. 15149–15154, 2001.

Rao, M. R. Cluster analysis and mathematical programming. *Journal of the American Statistical Association*, v. 66, n. 335, p. 622–626, 1971.

Resende, M.; Ribeiro, C. Greedy randomized adaptive search procedures: Advances and applications. In: Gendreau, M.; Potvin, J.-Y., eds. *Handbook of Metaheuristics*, v. (in press), 2nd ed, Springer, 2010.

Rezaee, R.; Lelieveldt, B.; Reiber, J. A new cluster validity index for the fuzzy c-mean. *Pattern Recognition Letters*, v. 19, p. 237–246, 1998.

Rice, S. A. The identification of blocks in small political bodies. *American Political Science Review*, v. 21, p. 619–627, 1927.

Romanowski, C. J.; Nagi, R.; Sudit, M. Data mining in an engineering design environment: OR applications from graph matching. *Computers & Operations Research*, v. 33, n. 11, p. 3150–3160, 2006.

Rosenwald, A.; Wright, G.; Chan, W. C.; Connors, J. M.; Campo, E.; Fisher, R. I.; Gascoyne, R. D.; Muller-Hermelink, H. K.; Smeland, E. B.; Giltnane, J. M.; Hurt, E. M.; Zhao, H.; Averett, L.; Yang, L.; M., H. E.; Zhao, H.; Averett, L.; Yang, L.; Wilson, W. H.; Jaffe, E. S.; Simon, R.; Klausner, R. D.; Powell, J.; Duffey, P. L.; Longo, D. L.; Greiner, T. C.; Weisenburger, D. D.; Sanger, W. G.; Dave, B. J.; Lynch, J. C.; Vose, J.; Armitage, J. O.; Montserrat, E.; López-Guillermo, A.; Grogan, T. M.; Miller, T. P.; LeBlanc, M.; Ott, G.; Kvaloy, S.; Delabie, J.; Holte, H.; Krajci, P.; Stokke, T.; Staudt, L. M. The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma. *The New England Journal of Medicine*, v. 346, p. 1937–1947, 2002.

Schaeffer, S. E. Graph clustering. *Computer Science Review*, v. 1, p. 27–64, 2007.

Shi, J.; Malik, J. Normalized cuts and image segmentation. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, p. 731–737.

Shi, J.; Malik, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, p. 888–905, 2000.

Shipp, M. A.; Ross, K. N.; Tamayo, P.; Weng, A. P.; Kutok, J. L.; Aguiar, R. C. T.; Gaasenbeek, M.; Angelo, M.; Reich, M.; Pinkus, G. S.; Ray, T. S.; Koval, M. A.; Last, K. W.; Norton, A.; Lister, T. A.; Mesirov, J. Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature Medicine*, v. 8, p. 68 – 74, 2002.

Su, A. I.; Cooke, M. P.; Ching, K. A.; Hakak, Y.; Walker, J. R.; Wiltshire, T.; Orth, A. P.; Vega, R. G.; Sapinoso, L. M.; Moqrich, A.; Patapoutian, A.; Hampton, G. M.; Schultz, P. G.; Hogenesch, J. B. Large-scale analysis of the human and mouse transcriptomes. *Proceedings of the National Academy of Sciences*, v. 99, p. 4465–4470, 2002.

Theodoridis, S.; Koutroubas, K. *Pattern recognition.* Academic Press, 1999.

Ushioda, A.; Kawasaki, J. Hierarchical clustering of words and application to nlp tasks. In: Ejerhed, E.; Dagan, I., eds. *Fourth Workshop on Very Large Corpora*, Somerset, New Jersey: Association for Computational Linguistics, 1996, p. 28–41.

van 't Veer, L. J.; Dai, H.; van de Vijver, M. J.; He, Y. D.; Hart, A. A.; Mao, M.; Peterse, H. L.; van der Kooy, K.; Marton, M. J.; Witteveen, A. T.; Schreiber, G. J.; Kerkhoven, R. M.; Roberts, C.; Linsley, P. S.; Bernards, R.; Friend, S. H. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, v. 415, n. 6871, p. 530–536, 2002.

Vinod, H. D. Integer programming and the theory of grouping. *Journal of the American Statistical Association*, v. 64, n. 326, p. 506–519, 1969.

Von Luxburg, U. A tutorial on spectral clustering. *Statistics and Computing*, v. 17, n. 4, p. 395–416, 2007.

Watts, D.; Strogatz, S. Collective dynamics of small-world networks. *Nature*, v. 393, p. 440, 1998.

Wei, Y. C.; Cheng, C. K. Towards efficient hierarchical designs by ratio cut partitioning. In: *Proceedings of the IEEE International Conference on Computer-Aided Design*, 1989, p. 298–301.

West, M.; Blanchette, C.; Dressman, H.; Huang, E.; Ishida, S.; Spang, R.; Zuzan, H.; Olson, J. A.; Marks, J. R.; Nevins, J. R. Predicting the clinical status of human breast cancer by using gene expression profiles. *PNAS*, v. 98, n. 20, p. 11462–11467, 2001.

White, S. D. M.; Frenk, C. S. Galaxy formation through hierarchical clustering. *Astrophysical Journal*, v. 379, n. Part 1, p. 52–79, 1991.

Wu, Z.; Leahy, R. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 15, n. 11, p. 1101–1113, 1993.

Xie, X.; Beni, G. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 13, p. 841–846, 1991.

Yeoh, E.-J.; Ross, M. E.; Shurtleff, S. A.; Williams, W. K.; D.Patel; Mahfouz, R.; Behm, F.; Raimondi, S. C.; Relling, M. V.; Patel, A.; Cheng, C.; Campana, D.; Wilkins, D.; Zhou, X.; Li, J.; Liu, H.; Pui, C.-H.; Evans, W. E.; Naeve, C.; Wong, L.; Downing, J. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer cell*, v. 1, p. 133–143, 2002.

Yu, S. X.; Shi, J. Multiclass spectral clustering. In: *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003, p. 313–319.