Eduardo Augusto Ribas

Clustering em Grafos

Curitiba

Eduardo Augusto Ribas

Clustering em Grafos

UNIVERSIDADE FEDERAL DO PARANÁ

Curitiba

Sumário

1 Introdução

Qualquer tipo de dados não uniformes contém uma estrutura subjacente devido a heterogeneidade dos dados. O processo de identificação dessa estrutura em termos de agrupar os dados em conjuntos é chamado *clustering*. Os grupos resultantes são chamados clusters. O agrupamento é normalmente baseado em alguma medida de similaridade definida sobre os dados. *Clustering* é intimamente ligado com aprendizagem não supervisionada em sistemas de reconhecimento de padrões, em que os dados são classificados em várias classes baseadas na medida de similaridade, sem ter nenhuma informação de como a classificação deve ser feita.

Grafos são estruturas formadas por um conjunto de vértices e um conjunto de arestas que são conexões entre os pares de vértices. *Clustering* em Grafos é a tarefa de agrupar os vértices do grafo em clusters levando em consideração a estrutura das arestas dos grafos de uma maneira que tenha muitas arestas dentro de cada cluster e, relativamente, poucas arestas fora de cada cluster. O campo de estudo desse problema tem se tornado popular e o número de algoritmos propostos para o problema é bem alto. Tal popularidade se deve à utilidade encontrada para o problema, principalmente para a área de mineração de dados.

Nesse texto será formalizado o conceito de *clustering*, identificando seus principais componentes, e enfatizando na difícil tarefa de definir boas medidas de similaridade. Na tentativa de encontrar bons resultados para o problema, diversas técnicas são utilizadas. A técnica que será utilizada neste trabalho se limita a técnicas de fluxo máximo e corte mínimo em grafos. Esta técnica é bastante utilizada pela complexidade dos algoritmos para resolver o problema do fluxo.

2 Notações

Nesta seção descreveremos algumas notações utilizadas.

No decorrer do texto, algumas convenções serão adotadas. O grafo será representado da forma usual, isto é, um grafo G é um par de conjuntos (V,E), onde V é o conjunto finito de vértices e E é um conjunto de arestas. Todo grafo mencionado será não dirigido.

Um cluster $c = \{v_1, v_2 \dots v_n\}$ é um subconjunto de vértices V de um grafo $G = (V,E)$. Uma partição de um grafo G é um conjunto $C = \{c_1, c_2 \dots, c_k\}$ de um grafo G, tal que a união de todos os clusters de $\cup_{c \in C} = V$, e a intersecção entre os clusters é vazia, ou, $a \cap b = \emptyset, \forall a, b \in C, a \neq b$.

2.1 Fluxo Máximo e Corte Mínimo

2.1.1 Fluxo Máximo

Supomos um material sendo conduzido dentro de um sistema em que existe uma fonte, onde o material é produzido, e um ralo, onde é consumido. A fonte produz o material de maneira contínua e sempre com a mesma taxa. O fluxo de um material em qualquer ponto do sistema é a taxa com que o material se move. Redes de fluxo podem ser usadas para modelar o escoamento de líquidos em canos, corrente elétrica percorrendo fios, informação entre redes de comunicação, redes viárias e assim por diante.

Ao modelar o mesmo problema em grafos, cada aresta do grafo pode ser pensada como um tubo de condução de um material. Cada tubo possui uma capacidade que pode ser interpretada como a maior taxa que um material pode escorrer pelo tubo, tal como 200 litros de água por minuto. Vértices são as junções dos tubos, e em cada vértice, excluindo a fonte e o ralo, a quantidade de material que sai é a mesma que entra em todo instante de tempo.

No problema de fluxo máximo, é necessário calcular a maior taxa com que cada material pode ser transportado da origem para o destino sem violar os limites de capacidade.

Seja $G = (V,E)$ com uma função de capacidade c. Seja s a fonte da rede, e t o ralo. Um fluxo em G é uma função $f : V * V \rightarrow R$ que satisfaz as três propriedades.

- Restrição de capacidade: Para todo $u, v \in V$, é preciso que $f(u,v) \leq c(u,v)$.
- Simetria: Para todo $u, v \in V$, é preciso que $f(u,v) = -f(v,u)$.
- Conservação do fluxo: Para todo $u \in V - s,t$, é preciso que $\sum_{v \in V} f(u,v) = 0$

A quantidade $f(u,v)$ é chamada de fluxo do vértice u até v. O valor de um fluxo f é definida como:

$$|f| = \sum_{v \in V} f(s,v)$$

Ou seja, é o fluxo total da fonte. No problema de fluxo máximo, é dado uma rede de fluxo G com fonte s e ralo t, sendo necessário achar o fluxo de maior valor.

O problema de fluxo máximo é um problema bem estudado na computação e possui vários algoritmos polinomiais para resolvê-los, com destaque ao algoritmo de Ford-Fulkerson [4].

2.1.2 Corte Mínimo

Um corte em um grafo é uma partição dos vértices do grafo em dois conjuntos. O conjunto de arestas de corte são as arestas que têm as pontas em dois conjuntos diferentes. O problema do corte mínimo consiste em, dado um grafo, achar um corte do grafo tal que a soma dos pesos das arestas de corte seja a menor possível. O problema de achar o corte mínimo entre dois vértices é similar, pois nele é preciso achar uma partição de vértices do grafo tais que os dois vértices estejam em conjuntos diferentes e a soma dos pesos das arestas de corte seja a mínima possível.

O teorema do fluxo-máximo corte-mínimo afirma que em uma rede de fluxos, o valor do fluxo máximo que passa da fonte para o ralo é igual ao corte mínimo entre os mesmos dois vértices. Pois o fluxo é a capacidade mínima quando as arestas são removidas de uma maneira específica e resultam que nenhum fluxo possa ir da fonte para o ralo, pois forma-se um corte que separa a fonte e o ralo. Assim, é possível calcular o corte mínimo de um grafo a partir do fluxo máximo, desfrutando dos algoritmos polinomiais para resolver o problema do fluxo.

Dessa forma, dois conjuntos de vértices A e B de um grafo $G = (V,E)$, tal que $A \in V$ e $A \cap B$, definem um corte em G, que será denotado por (A,B). A soma de arestas que cruzam o corte determinam o valor do corte, que será denotado por $c(A,B)$.

2.2 Árvore de corte mínimo

Seja $G = (V,E)$ um grafo. Se for preciso computar o corte mínimo global do grafo, isso pode ser facilmente calculado aplicando algoritmos de fluxo máximo/corte mínimo, como descrito acima, para todos os $\binom{|V|}{2}$ possíveis pares de vértices. O problema para esta abordagem é que processar todas essas operações pode ser muito custoso. Gomory e Hu [6] mostraram que o número de cortes mínimos distintos entre dois vértices de um grafo é no máximo $|V|-1$, e existe uma estrutura de árvore eficiente, a árvore de corte mínimo, que pode ser construída com no máximo $|V|-1$ aplicações do algoritmo de fluxo. Após construída, com a ajuda da árvore de corte mínimo, é possível realizar consultas de corte mínimo entre pares de vértices em tempo apenas $O(|V|)$. Apesar de que o problema de corte mínimo global possui algoritmos mais eficientes para resolvê-lo, a árvore de corte mínimo tem um propósito maior e consegue resolver uma gama maior de problemas.

Seja $G = (V,E)$ um grafo não direcionado, e um custo associado a cada aresta, também chamado de peso da aresta, representado por uma função $c : E \to \mathbb{R}$. Denotaremos a capacidade do corte mínimo entre s e t por $\lambda_{st}, \forall s,t \in V$.

Seja $T = (V_T, E_T)$ uma árvore com $V_T = V$, e P_{st} o conjunto de arestas no caminho $s-t$ em T, $\forall s,t \in V_T$

Então T é uma árvore de corte mínimo de G se:

$$\lambda_{st} = \text{Min}_{e \in P_{st}} c(S,T), \forall s,t \in V$$

onde:

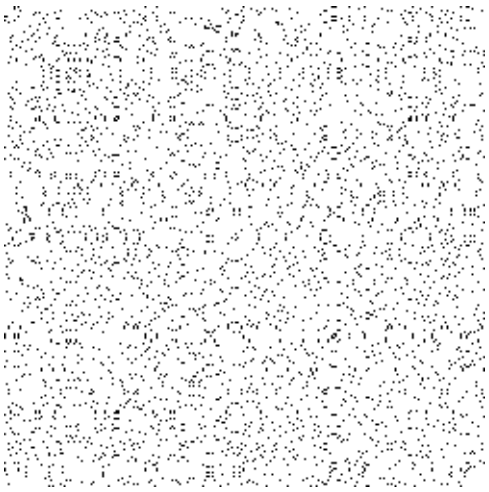
- S e T são os conjuntos de vértices resultantes do corte $s-t$
- $c(S,T)$ a capacidade do corte em G

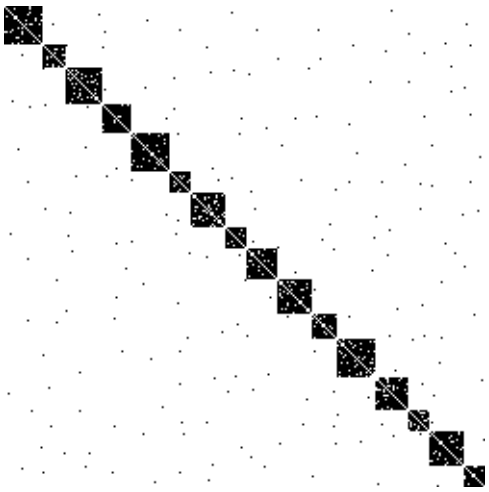
A árvore de corte mínimo é definida sobre V e tem a propriedade de que é possível achar o corte mínimo entre dois vértices s e t em G apenas inspecionando o caminho que conecta s e t em T_G. Como T_G é uma árvore, há apenas um caminho ligando s e t, e aresta de capacidade mínima nesse caminho corresponde ao corte mínimo. A remoção dessa aresta resulta em dois conjuntos de vértices em T_G, que são iguais aos dois conjuntos de vértices do corte mínimo entre s e t em G. Para todo grafo não direcionado, sempre existe uma árvore de corte mínimo

3 O problema

Dado um conjunto de dados, o objetivo do problema de *clustering* é particionar o conjunto de dados em clusters ou agrupamentos tais que os elementos designados para um cluster são similares ou conectados de alguma forma predefinida. Entretanto, nem todos os conjuntos possuem uma estrutura natural com clusters. Mesmo assim, um algoritmo de *clustering* retorna um agrupamento para qualquer grafo dado a ele. Se a estrutura do grafo for totalmente uniforme, com arestas uniformemente distribuídas entre os vértices, o agrupamento computado por qualquer algoritmo será arbitrário. Medidas de qualidade, e se possível, visualizações, ajudam a determinar se há um cluster significativo presente no grafo e se um dado agrupamento os revelam.

Para dar uma ideia mais concreta do que são os clusters, é apresentado aqui um pequeno exemplo na figura abaixo. Na figura apresentada, nós temos a representação da matriz de adjacência de um grafo, em que um ponto na linha i e coluna j é preto se o vértice i possui uma ligação com o vértice j, e branco caso contrário.





O grafo representado pela figura acima possui 250 vértices e 3674 arestas. Quando os vértices estão dispostos em um ordem aleatória, não existe uma estrutura subjacente aparente na matriz de adjacência e não é possível interpretar trivialmente a presença, número ou qualidade

dos clusters inerentes no grafo. Porém, quando os vértices são organizados de uma outra forma, através de um algoritmo de *clustering* por exemplo, é obtida uma versão diagonalizada da matriz de adjacência, mostrada no lado direito da figura. Agora a estrutura com clusters é evidente: temos 16 clusters densos de ordens variadas e algumas conexões esparsas entre os clusters.

A pergunta que vem agora é saber o que são bons clusters. Um particionamento tem resultado bom se os vértices que são similares são designados para o mesmo cluster e pontos que são dissimilares, designados para diferentes clusters. Existem várias estratégias que tentam traduzir o quão similar um ponto é com o outro. Agora iremos nos restringir a uma simples ideia, dois pontos são similares caso exista uma aresta que os ligam, e são dissimilares caso contrário. Nesse caso, é possível perceber que nem sempre é possível ter um particionamento perfeito, ou seja, nem sempre é possível que todos os pontos similares sejam designados para clusters iguais.

A maneira mais trivial de se pensar para saber o quão bom é um particionamento, seria comparar um particionamento gerado com o particionamento ótimo do grafo. Porém, não sabemos qual é o particionamento ótimo para um grafo dado, muito menos o que significa um particionamento ótimo. Para dar significado ao particionamento ótimo, precisamos dar uma resposta quantitativa. Para tanto é preciso definir medidas de qualidade de um particionamento, assim será possível quantificá-lo. Tendo medidas de qualidade é possível definir o problema de *clustering*, pois com ela, o objetivo do problema de *clustering* seria achar uma partição que a maximize.

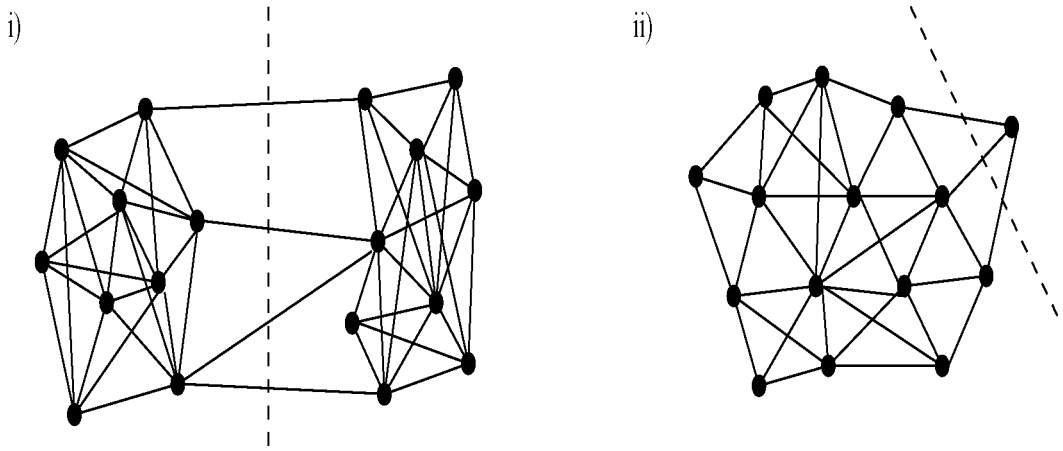
Com todas esses pensamentos em mente, podemos formalizar o problema envolvido em *clustering* da seguinte forma. Seja $P(C)$ o conjunto de partições de um conjunto C. Podemos definir *clustering* como o problema de, dado um conjunto finito de objetos C e uma função de qualidade $f : P(C) \to Q$, encontrar $x \in P(C)$ tal que $f(x)$ seja máximo.

Olhando para essa definição podemos ver que todos os elementos discutidos anteriormente estão presentes nela, pois antes de quantificar os clusters, discutimos como particionar e o que era um bom particionamento. Se forem criadas medidas que traduzam o que as ideias intuitivas falam, teremos exatamente o problema que discutíamos.

4 Medidas

4.1 Corte mínimo

Antes de mais nada, iremos retornar a questão de o que é um bom cluster. A qualidade de um cluster deve ser determinada pela similaridade dos vértices dentro de um cluster. Em particular, se existe um corte de custo pequeno que divide um cluster em dois conjuntos de tamanho similar, então o cluster possui vários pares de vértices que são dissimilares e, portanto, é de baixa qualidade. Isso sugere que a qualidade de um cluster é o corte mínimo de um cluster. Porém essa afirmação pode ser facilmente derrubada pela figura abaixo.



Nestes exemplos, podemos perceber dois problemas com a medida de corte mínimo. No parágrafo anterior, foi dito que se existe um corte de custo pequeno que divide um cluster em dois conjuntos de tamanho similar, então o cluster é ruim. Um dos problemas é o tamanho dos conjuntos que o corte mínimo gera. Em muitos casos os conjuntos gerados possuem tamanho bem desigual, o que não necessariamente representa que o cluster possui muitos elementos dissimilares, supondo que uma aresta entre dois vértices representa um par que tem uma alta similaridade.

O segundo problema da medida de corte mínimo é em relação ao valor: um cluster bom

é um que contém o valor do corte mínimo baixo. Porém não foi dito em momento algum o critério que é usado para dizer se um corte mínimo tem valor baixo. Se não é usado critério algum, temos um problema bem claro representado na figura. O corte mínimo do primeiro grafo é maior que o segundo, devido ao fato de que o segundo grafo possui vértices de grau baixo. No entanto, o segundo grafo é um cluster de maior qualidade, mesmo tendo um corte mínimo menor do que o primeiro. Isso porque o primeiro grafo possui um corte cujo peso é pequeno em relação aos tamanhos dos subconjuntos que ele cria. A medida que quantifica a relação do valor do corte com seus subconjuntos formados pelo corte é a expansão que será descrita a seguir.

4.2 Expansão

Seja $(S,\overline{S})$ um corte em G. Definimos a *Expansão* [7] de um corte como:

$$\psi(S,\overline{S}) = \frac{\sum_{u\in S, v\in \overline{S}} w(u,v)}{min\{|S|,|\overline{S}|\}}$$

Podemos definir a *Expansão* de um grafo $G = (V,E)$ como sendo a menor expansão dentre todos os cortes de G, ou seja:

$$\psi(G) = \mathrm{Min}_{S\in V} \frac{\sum_{u\in S, v\in \overline{S}} w(u,v)}{min\{|S|,|\overline{S}|\}}$$

Com a medida de Expansão bem definidas, podemos estender essa definição para a Expansão de uma Partição de G. Dado uma partição C de um grafo G, podemos definir a Expansão de uma partição de clusters como sendo a menor Expansão dentre suas partições.

$$\Psi(C) = \mathrm{Min}_{S\in C} \psi(S)$$

Como podemos perceber, a medida da expansão fornece igual importância a todos os vértices do grafo. Isso, porém, pode ainda gerar alguns problemas, assim como o problema da medida do corte mínimo. Como a expansão dá igual prioridade a todos seus vértices, existem cenários em que alguns vértices que representam uma pequena porção do grafo que possuem poucas arestas entre si e poucas arestas com o grafo todo, podem puxar para baixo o valor da expansão de um grafo inteiro, mesmo sendo um bom cluster.

A partir disso podemos perceber que é mais prudente dar maior importância a vértices que tem grau alto e pouca importância a vértices que tem grau baixo. Podemos generalizar essa

ideia acima para dar espaço à condutância, que é a medida que veremos abaixo.

4.3 Condutância

Seja $(S,\overline{S})$ um corte em G. Definimos a condutância [7] de um corte como:

$$\varphi(S,\overline{S}) = \frac{\sum_{u\in S, v\in\overline{S}} w(u,v)}{min\{c(S),c(\overline{S})\}}$$

Estendendo a medida de condutância de um corte para um grafo, temos:

$$\phi(G) = \mathrm{Min}_{S\in V} \frac{\sum_{u\in S, v\in\overline{S}} w(u,v)}{min\{c(S),c(\overline{S})\}}$$

Dada uma partição C de um grafo G, podemos definir a Condutância de uma partição de clusters como sendo a menor Condutância dentre os clusters.

$$\Phi(C) = \mathrm{Min}_{S\in C}\phi(S)$$

Assim como a *Extensão*, a condutância é definida da mesma maneira sobre um corte, grafo e sobre uma partição de um grafo.

A condutância parece ser uma medida muito boa para o nosso problema de *clustering*, ou seja, podemos usar a função de qualidade da definição do problema de *clustering* como sendo a condutância.

4.4 Análise

A *Expansão*, tanto como a *Condutância* resultam em boas medidas de qualidades para o problema de *clustering*, e são, na maioria das vezes, consideradas melhores que cortes mínimos. A principal diferença entre as duas medidas é que a *Expansão* considera todos os vértices igualmente importantes enquanto a *condutância* dá uma importância maior para os nós com alto grau.

Ambas as funções são também parecidas em relação à sua definição. Quanto menor a *Condutância* ou *Expansão* de um corte, a proporção das arestas de corte em relação ao grafo será pequena, resultando em clusters mais densos que o grafo original.

Já em relação às medidas relativas ao grafo, que são definidas como o menor valor de

medida para todo corte do grafo, a lógica de quem é melhor é o inverso do corte. Pensando nessas medidas como medidas intra-clusters, ou seja, medidas que definem a qualidade do grafo, quem tiver a maior *Condutância* ou *Expansão* são grafos de melhor qualidade. Isso se deve ao fato da própria definição, pois se um grafo tiver uma *Condutância* ou *Expansão* baixa, significa que ele possui um corte com um valor baixo da medida correspondente, podendo inferir que o grafo possui muitos elementos dissimilares e pode ser particionado gerando clusters de uma qualidade ainda maior. Por outro lado, se a medida de qualidade de um grafo for alta, significa que todos os cortes do grafo geram medidas altas, ou seja, o grafo é bem coeso e possui grande similaridade entre todos seus vértices.

Quanto a qualidade de um particionamento em relação às medidas, é análoga à qualidade de um grafo em relação à uma medida. A medida de um cluster, ou subgrafo, é boa, se ela tem um valor alto, como discutido anteriormente. Se uma partição tem um valor de medida alto significa que todos seus clusters têm valores de medida maiores ou iguais ao da partição, e portanto todos clusters possuem valores grandes. Se uma partição tem valor baixo, significa que algum cluster possui um valor de medida baixo, e consequentemente tem uma qualidade baixa.

Apesar das medidas apresentarem parecerem intuitivamente boas, ainda há problemas com elas. Um grafo pode consistir em sua maioria de clusters de alta qualidade e talvez alguns clusters de qualidade muito baixa, de tal forma que qualquer partição necessariamente tem uma qualidade global ruim. Isso é devido a forma de como são definidas as medidas de qualidade em relação uma partição. Então, para melhorar a qualidade global, a melhor partição provavelmente seja criar muitos clusters de relativamente baixa qualidade tal que a menor qualidade seja mais alta possível. Isso implica que a partição teoricamente ótima fique mascarada devido a alguns clusters ruins. Ainda que existam problemas com a medida da condutância, calculá-la não é uma tarefa trivial, pois o problema de achar um corte em um grafo G tal que $\phi(G) \leq \rho$, para um inteiro positivo ρ qualquer , é $\mathbb{NP}$-completo [12].

4.5 Similaridade entre Partições

Caso for necessária a comparação entre diferentes partições, é preciso também ser definido uma medida para isso, para que seja possível quantificar a diferença ou similaridade entre duas partições. Para isso, é definida a medida $\mu_\sigma(C,S)$ entre duas partições C e S do mesmo grafo.

$$\mu_\sigma(C,S) = \frac{\dfrac{\sum_{c\in C} max_{s\in S}\dfrac{|c\cap s|}{|c\cup s|}}{|C|} + \dfrac{\sum_{s\in S} max_{c\in C}\dfrac{|s\cap c|}{|s\cup c|}}{|S|}}{2}$$

O maior valor que essa medida pode assumir é 1, que ocorre quando as duas partições são iguais. Quanto maior o valor da medida é, mais similares são as duas partições. Esta medida é bastante útil quando quer comparar o resultado de uma partição de algum algoritmo com uma partição desejada.

5 *O algoritmo de* CutClustering

5.1 Ideia

Na seção 4 descrevemos algumas ideias de bons particionamentos e a principal entre elas era que dois vértices em um grafo são similares se eles possuem uma aresta ligando-os ou são dissimilares caso contrário. O algoritmo descrito aqui usa ideias baseadas em fluxo, ou seja, se o fluxo entre dois vértices forem relativamente altos, então os dois vértices são similares. Pela relação que existe entre fluxo máximo e corte mínimo, a frase anterior é equivalente se trocar fluxo por corte mínimo, cuja ideia fica parecida com a da medida de corte mínimo discutida anteriormente.

Podemos denotar o conjunto que se forma por um corte (s,t) em que s está contido por uma comunidade de s em relação a t. No trabalho de Flake[3] foi definido uma *comunidade web*. Uma *comunidade web* é um conjunto de vértices tal que os vértices da comunidade web predominantemente ligam entre si do que o resto do grafo. Em outras palavras, seja S um conjunto de vértices e uma comunidade web:

$$\sum_{v \in S} w(u,v) \quad > \quad \sum_{v \in \overline{S}} w(u,v), \forall u \in S \tag{5.1}$$

As comunidades web têm uma forte relação com o corte mínimo, pois seja S a comunidade de s em relação a t, S é uma comunidade web. Isto é facilmente verificado, pois todo vértice $w \in S - s$ tem pesos maiores em S do que para vértices em $V - S$, caso contrário, movendo w para o outro lado do corte, resultaria em um corte menor entre s e t.

Seja um grafo G e um inteiro $k > 1$, o problema de particionar G em k comunidades web é $\mathbb{NP}$-completo. [3]

Voltando para a ideia do algoritmo e ainda pensando no problema do corte mínimo, em que é dado dois vértices s e t para se calcular o corte mínimo, ou a comunidade de s em relação a t. O ideal para o problema de *clustering* seria calcular a comunidade de um vértice s sem a

relação a um outro vértice. Para tanto chegamos a seguinte questão: E se tirarmos t da definição de comunidade?

A tentativa que [2] usou para tentar resolver esse problema é simples. É adicionado um vértice adicional t ao grafo, que é conectado a todos os outros vértices do grafo com peso α. Assim quando quisermos nos referir a uma comunidade de um vértice v qualquer, basta calcular a comunidade de v em relação ao vértice adicional t.

Agora nos resta a análise para saber quem são os elementos que são designados para a comunidade de um vértice s. Esta questão será discutida mais detalhadamente na próxima seção, porém já adiantarei a ideia principal. Um vértice s terá em sua comunidade o vértice u se for mais fácil ir de s para u sem passar por t. O termo fácil se refere ao fluxo. Ou seja, se as arestas que passam por t influenciarem muito no fluxo entre s e u, então eles estarão em clusters diferentes, caso contrário ficarão no mesmo cluster.

5.2 Teorema Principal

A parte mais interessante do algoritmo aqui apresentado para resolver o problema de *clustering*, é a garantia dos resultados gerados. Muitos algoritmos de *clustering* se baseiam em heurísticas e não comprovam que tipo de resultado que o algoritmo gera, apenas afirmam que o resultado gerado pode ser bom para determinados tipos de grafo. No caso do *CutClustering*, dependendo de α escolhido para ligar o vértice adicional, o resultado gerado tem uma desigualdade bem definida.

Seja S a comunidade de um vértice s qualquer, então:

$$\frac{c(S,V-S)}{|V-S|} \leq \alpha \leq \psi(S) \tag{5.2}$$

Podemos analisar o primeiro termo da desigualdade acima como uma medida parecida com a expansão do corte entre S e $V-S$, só que seu valor pode ser menor do que a expansão, pois $|V-S| \geq min(|V-S|,|S|)$, e portanto $\frac{1}{|V-S|} \leq \frac{1}{min(|V-S|,|S|)}$, e assim $\frac{c(S,V-S)}{|V-S|} \leq \frac{c(S,V-S)}{min(|V-S|,|S|)}$. Sabemos que a expansão de um corte de um cluster em relação ao resto do grafo tem que ser preferencialmente baixa, então valores baixos de α criam clusters com medidas inter-clusters boas.

O último termo é exatamente a expansão de S, e sabemos que quanto maior a expansão de um cluster, melhor ele é. Então, para termos clusters com alta expansão, α também deverá ser grande. Assim podemos perceber que a qualidade dos clusters gerados dependem muito do

valor de α. Assim não podemos escolher valores de α muito baixos para termos uma expansão alta, nem valores muito altos para ter a medida de corte alta.

Para provar essa desigualdade, primeiro será provado os dois primeiros termos da desigualdade e em seguida os dois últimos termos.

Lema 1. *Seja G_α o grafo expandido de $G = (V,E)$ e seja S a comunidade de s em relação ao ralo artificial t, então o limite sempre vale:*

$$\frac{c(S,V-S)}{|V-S|} \leq \alpha$$

Demonstração. Seja T_{G_α} a árvore de corte mínimo de G_α, e t o vértice adicional. O corte mínimo entre s e t em T_{G_α} é um conjunto de apenas uma aresta (s',t') de T_{G_α} cuja remoção gera os conjuntos S e $V-S\cup\{t\}$. Considere também o corte $(V,\{t\})$ em $\{t\}_{G_\alpha}$, que também separa s de t, porém o valor desse corte é maior ou igual ao corte mínimo, então:

$$c(S,V-S\cup\{t\}) \quad \leq \quad c(V,\{t\})$$

Para desenvolver essa fórmula, nos atentemos ao seguinte fato. Seja um corte qualquer $c(S,P\cup Q)$, em que $P\cup Q = V-S$, então $c(S,P\cup Q) = c(S,P)+c(S,Q)$. Ou seja, o corte entre dois conjuntos P e Q com um conjunto S é igual ao corte de $P\cup Q$ e S. Pois por definição, o corte entre dois conjuntos são as arestas que tem pontas em conjuntos diferentes. O corte entre P e S são as arestas que tem ponta em S e a outra ponta em P, e assim analogamente a P e Q. Então a união entre esses dois cortes são as arestas que tem ponta em S e a outra ponta em P ou Q, que é exatamente o corte entre S e $P\cup Q$.

Usando essa ideia para separar t da primeira expressão, e para dividir a segunda expressão, temos:

$$c(S,V-S)+c(S,\{t\}) \quad \leq \quad c(V-S,\{t\})+c(S,\{t\})$$

Cancelando os dois termos iguais, temos:

$$c(S,V-S) \quad \leq \quad c(V-S,\{t\})$$

Sabemos que t liga a todas arestas do grafo com peso α, então sabemos de antemão qual o valor de $c(V-S,\{t\})$, que é justamente $\alpha|V-S|$.

$$\begin{aligned} c(S,V-S) &\leq \alpha|V-S| \\ \frac{c(S,V-S)}{|V-S|} &\leq \alpha \end{aligned}$$

□

Lema 2. *Seja G_α o grafo expandido de $G=(V,E)$ e seja S a comunidade de s em relação ao ralo artificial t. A expansão de S ou $\phi(S)$, ou seja, para qualquer conjunto não vazio P e Q, tal que $P \cup Q = S$ e $P \cap Q = \emptyset$, então o limite sempre vale:*

$$\alpha \leq \frac{c(P,Q)}{min(|P|,|Q|)}$$

Demonstração. Seja T_{G_α} a árvore de corte mínimo de G_α, e t o vértice adicional. Seja (s',t') uma aresta de T_{G_α} cuja remoção resulta em S e $V-S\cup\{t\}$ em G_α. De acordo com as propriedades da árvore de corte mínimo, (s',t') existe e está no caminho de (s,t) em T_{G_α}

Agora Dividimos o conjunto S em dois subconjuntos P e Q, e assumindo que $s' \in P$, então:

$$c(V-S\cup\{t\},Q) \leq c(P,Q)$$

A prova de corretude dessa expressão vem da própria definição da árvore de corte mínimo. Se algum corte para qualquer conjunto $P,Q \in S$ em T_{G_α} fosse menor que $c(V-S\cup\{t\},Q)$, então a aresta de custo mínimo de T_{G_α} que separaria s de t estaria completamente em S, só que pela enunciação, já foi pega a aresta de menor custo que separa s de t. Então de fato $c(V-S\cup\{t\},Q) \leq c(P,Q)$.

Com o mesmo raciocínio da outra prova, separaremos $\{t\}$ de $c(V-S\cup\{t\},Q)$, e retiraremos a primeira parcela da soma sem ferir a desigualdade:

$$\begin{aligned} c(V-S,Q)+c(\{t\},Q) &\leq c(P,Q) \\ c(\{t\},Q) &\leq c(P,Q) \end{aligned}$$

O corte $c(\{t\},Q)$ é exatamente $\alpha * |Q|$, pois t é ligado com todos os vértices do grafo com arestas de peso α, então:

$$\alpha|Q| \leq c(P,Q)$$

Ainda sem ferir a desigualdade, podemos substituir $|Q|$ por $min(|P|,|Q|)$, pois $min(|P|,|Q|) \leq |Q|$ e assim, $\alpha.min(|P|,|Q|) \leq \alpha|Q| \leq c(P,Q)$

$$\begin{aligned} \alpha.min(|P|,|Q|) &\leq c(P,Q) \\ \alpha &\leq \frac{c(P,Q)}{min(|P|,|Q|)} \end{aligned}$$

□

5.3 Algoritmo

Usando as ideias descritas, podemos construir o algoritmo *CutClustering*. Para construí-lo precisamos saber a comunidade de cada vértice do grafo em relação a um determinado α. Porém, o conjunto de todas as comunidades não é uma partição, pois pode haver intersecção entre as comunidades de vértices. Analisando a árvore de corte mínimo, podemos perceber que a intersecção entre as comunidades de 2 vértices a e b ou é vazia, ou uma comunidade é o subconjunto da outra. Para fins de particionar, pegaremos todas as comunidades que não são subconjuntos de nenhuma outra. Os vértices que possuem esta propriedade são os vértices adjacentes a t na árvore de corte mínimo. E assim serão pegas as comunidades desses vértices, e sua união forma uma partição.

Algorithm 1 CutClustering$(G(V,E),\alpha)$

$V' \leftarrow V \cup t$
for all vertices v em V **do**
  Conecta v e t com peso α
end for
$G'(V',E') \leftarrow$ grafo estendido após conectar t com V
$T' \leftarrow$ árvore de corte mínimo de G'
Remove t de T'
return todas as componentes conexas como clusters de G

5.4 Escolhendo α

Agora que sabemos quais as propriedades dos resultados podem surgir dependendo da escolha do α. A escolha do α tem um papel crucial na qualidade dos clusters gerados. Mas como escolher o melhor valor de α?

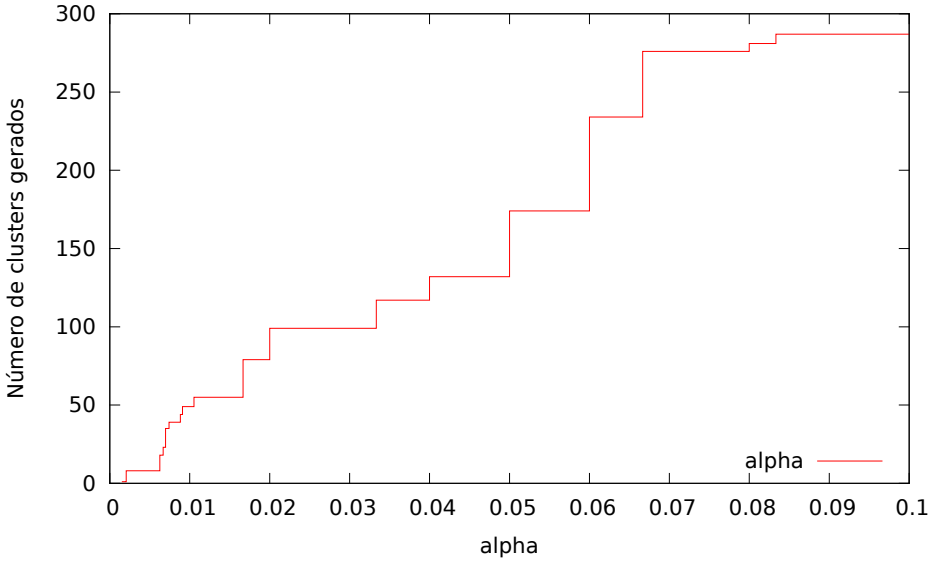
É fácil de ver que a medida que α se aproxima de 0, o corte mínimo entre o vértice adicional t e o resto do grafo irá se tornar o corte trivial (t,V), em que isola t do resto dos vértices, e de acordo com o algoritmo, gera apenas um cluster que é o grafo inicial G por inteiro. Por outro

lado, a medida que α tende ao infinito, T_{G_α} irá tender a ser um grafo estrela, ou seja, um grafo com o vértice t no centro e todos os outros vértices ligados somente a t. Assim, de acordo com o algoritmo de *CutClustering*, será produzido $|V|$ clusters, em que cada um deles terá apenas um vértice.

Podemos definir alguma estratégia de escolha para o α. Testar todos os possíveis valores α não é uma boa ideia, pois α pode pertencer aos números reais, que é um conjunto infinito e extremamente denso. Porém, α tem uma propriedade muito importante, que é intuitivamente descrita no parágrafo anterior. Se α tende a 0, então ele gera apenas 1 cluster. A medida que α cresce, o número de clusters gerados aumenta, e os clusters novos gerados são uma união dos clusters antigos com valores de α menor. Formalizando esta ideia temos:

Para uma fonte s em G_{α_i}, onde $\alpha_i \in \{\alpha_1, \ldots, \alpha_{max}\}$ tal que $\alpha_1 < \alpha_2 \cdots < \alpha_{max}$, as comunidades $S_1, \ldots, S_{max}$ são tais que $S_1 \subseteq S_2 \subseteq \ldots \subseteq S_{max}$, onde S_i é a comunidade de s em relação a t em G_{α_i}. [11, 5]

Para termos uma ideia da facilidade que isso traz ao nosso problema, o gráfico abaixo mostra o número de clusters gerados para valores de α diferentes de um grafo.



Sabemos que, em cada nível do gráfico (linha horizontal), além do número de clusters gerados ser o mesmo, as partições em si também são as mesmas. Como temos uma função não decrescente e bem comportada, podemos aplicar uma variação da busca binária para achar todos os valores de α que geram partições diferentes. E como o número de partições diferentes geradas são no máximo $|V| - 1$ [11, 5], essa abordagem é viável.

Com todas as partições possíveis geradas pelo algoritmo, podemos escolher a partição que

tenha o maior valor em relação à medida de qualidade escolhida para resolver o problema. O mais importante é que as partições geradas não são quaisquer partições, mas estão dentro do limite do α.

6 Implementação

Sabemos que α pode assumir valores reais, e não apenas inteiros. Porém a representação de números reais em computadores pode ter problemas de precisão e a performance de alguns algoritmos de fluxo pode cair drasticamente por conta disso. Assim, para evitar o uso de valores ponto flutuante, foram usados inteiros de 64 bits e todas as arestas do grafo foram multiplicadas por uma constante alta, não alterando as propriedades do grafo. Com isso, todos os valores que α pode assumir são todas as frações com denominador determinado pela constante e o numerador um inteiro de 64 bits qualquer.

Sabe-se que existem vários algoritmos para se calcular o fluxo máximo, mas o escolhido foi o algoritmo de Ford-Fulkerson[4]. Este não é o algoritmo mais eficiente dentre os algoritmos de fluxo, porém seu funcionamento é bem simples e pode ser bem eficiente em grafos esparsos. Para uma melhor eficiência do algoritmo de *Ford-Fulkerson*, é necessária a representação do grafo por uma lista de adjacência, e assim o algoritmo fica mais eficiente para grafos esparsos.

Um passo importante do algoritmo de *CutClustering* é a construção da árvore de corte mínimo. Sabemos, pela descrição do algoritmo, que a árvore de corte mínimo é construída e um vértice é desconectado dela e pegam-se as componentes conexas. Em outras palavras, a construção da árvore é desconsiderada para todos os vértices que não são adjacentes a t em T_G. Assim, como a construção da árvore pelo algoritmo de *Gomory-Hu* é por blocos, ou seja, começa-se com um bloco inicial que é o grafo inteiro, particiona-se e processa-se independentemente cada bloco resultante, quando um bloco não cotém o vértice adicional t, seu processamento não precisa ser continuado.

7 Benchmarking

Um problema geral relacionado à avaliação de algoritmos de *clustering* é a falta de um conjunto de dados com uma estrutura subjacente já conhecida. Essa falta tem raízes na indefinição do problema de *clustering*, pois a melhor partição de um conjunto de dados depende muito de sua aplicação, e a estrutura subjacente ótima de sua aplicação. Um grande problema relacionado às medidas é achar a partição ótima que maximiza a medida. Porém muitas medidas tornam o problema de *clustering* $\mathbb{NP}$-difícil. Assim sendo, criar um conjunto de dados genérico e já particionado otimamente se torna uma tarefa difícil.

Apesar disso, como discutido anteriormente, as medidas foram criadas com base em intuições humanas de bons particionamentos. Então, ao invés de obter respostas ótimas para os grafos de um conjunto de dados, criam-se grafos previamente particionados, ou grafos artificiais, com base nas ideias intuitivas de bons particionamentos, ou seja, as ligações intra-clusters são altas e as ligações inter-clusters são baixas.

A grande maioria das aplicações dos problemas de *clustering* não requerem uma resposta ótima para o problema, apenas boas aproximações são o necessário, e em muitos casos, a velocidade é um fator ainda mais decisivo. Então existem vários fatores para se determinar um bom algoritmo de *clustering*.

7.1 Grafos Artificiais

Antes de partir para experimentos e geração de grafos artificiais, é preciso formalizar o que é um grafo artificial, pois apenas a ideia de ligações intra-cluster e inter-cluster não são suficientes. Porém, essa é a ideia principal dos grafos artificiais, em que há uma probabilidade z_{in} de ocorrer arestas intra-clusters, e probabilidade z_{out} de ocorrer arestas inter-clusters. E para ter uma estrutura em forma de clusters, $z_{in} > z_{out}$.

Em [8] são citadas várias definições das principais componentes dos grafos artificiais. Para definí-las, antes definiremos outros conceitos mais simples.

Sabemos que uma partição de um grafo é um conjunto de clusters $C = c_1, c_2, c_3, \ldots, c_k$. Podemos denotar por $c(v)$ o cluster c_i ao qual o vértice v pertence. E finalmente denotamos $g(c)$ o subgrafo de G induzido pelo cluster c.

O acoplamento entre um cluster c e um vértice v é denotado por $l(c,v)$ e é igual ao número de arestas de v que terminam em um vértice de c.

Uma partição C de um grafo é chamado de tradicional se o acoplamento entre um vértice e seu cluster for maior que o acoplamento entre o vértice e o resto do grafo. Ou seja:

$$l(c(v),v) > l(V\text{–}c(v),v), \forall v \in V$$

Uma partição C de um grafo é chamada de detalhada se o acoplamento entre um vértice e seu cluster for maior que o acoplamento entre o vértice e qualquer outro cluster, ou seja:

$$l(c(v),v) > l(c,v), \forall v \in V, c \in C\text{–}c(v)$$

A densidade do acoplamento interno de um vértice v, é igual ao acoplamento entre v e $c(v)$ dividido pelo tamanho de $c(v)-1$, ou seja:

$$ild(v) = \frac{l(c(v),v)}{(|c(v)|\text{–}1)}$$

A densidade interna de um grafo particionado $id(G,C)$ é igual a média da densidade do acoplamento interno de todos os vértices.

$$id(G,C) = \frac{\sum_{v \in V} ild(v)}{|V|}$$

E, finalmente, definimos um grafo otimamente particionado como sendo um grafo detalhado ou tradicional e que satisfaz as seguintes condições:

$$\{v,v\} \notin E, \forall v \in V$$

Esta primeira condição apenas afirma que um grafo não pode ter uma aresta ligando um vértice a ele mesmo.

$$g(c) \text{ é conexo }, \forall c \in C$$

Esta segunda condição é a mais crucial para o grafo artificial. Pelas ideias iniciais de bons clusters, um cluster jamais será desconexo. Essa condição influencia na geração dos grafos artificiais, pois um algoritmo ingênuo poderá não garantir essa condição.

7.2 Construção de grafos artificiais

Para se realizar um benchmark é necessário a criação de vários grafos com densidades diferentes. Para tanto, o processo de criação desses grafos deve ter um alto grau de importância e garantir que os grafos sejam corretamente gerados.

Sabemos que as duas principais componentes dos grafos artificiais são a densidade interna e a densidade externa. O processo de criação do grafo será dividido em duas partes. A primeira é a criação de um grafo base, que se encarrega de garantir a densidade interna, e a conectividade dos clusters. A segunda visa garantir a densidade externa e a validação de um grafo otimamente particionado.

7.2.1 Grafo Base

Um grafo é chamado de grafo base se ele é otimamente particionado e o conjunto de arestas inter-cluster é vazio. Ou seja, um grafo base é, na verdade, apenas um conjunto de clusters com uma densidade interna média definida por um valor.

Portanto, o grafo base tem quatro componentes principais: um número de clusters m; o número mínimo e máximo de vértices por clusters, e um número real r representando a densidade interna.

Após ter feito a tarefa trivial de criar os m clusters com seus respectivos vértices, é preciso adicionar as arestas e garantir que cada cluster é conexo, e que a densidade interna seja alcançada. Existem várias maneiras de construí-lo, mas a maneira aqui escolhida foi a que consegui garantir as restrições mais facilmente. Assim, envés de começar com um grafo sem arestas, começam-se com m cliques, e retiram-se arestas até atingir a densidade interna desejada, tomando cuidado para não retirar arestas que tornam cada cluster desconexo.

7.2.2 Grafo Otimamente Particionado

A segunda parte do processo de criação dos grafos artificiais, é a criação dos grafos otimamente particionados, tendo já pronto um grafo base. Nessa fase é necessária a adição de arestas

inter-clusters de tal forma que tenha-se uma densidade desejada e seja satisfeita a condição de grafo otimamente particionado, ou seja, ou é um grafo detalhado ou é tradicional.

A densidade que será utilizada é a taxa de acoplamento transicional, que é a razão entre número de arestas inter-clusters dividido pelo número de arestas intra-clusters.

Para garantir essas propriedades, usaremos uma estrategia inversa a usada no grafo base, adicionaremos arestas basta adicionar um número de arestas aleatoriamente até que tenha a densidade desejada. Tendo cuidado que uma aresta só é adicionada se não infrigir a restrição do grafo ser tradicional ou detalhado.

8 Testes

8.1 experimentos

Para a realização de testes em cima de um programa que implementa um determinado algoritmo, como sugerido em [8], é necessária a criação de vários grafos com densidades diferentes.

Para este trabalho foram gerados alguns grafos otimamente particionados, variando-se o número de arestas intra-cluster e inter-cluster, porém mantendo a configuração do número de vértices e clusters do grafo base, para melhor compará-los.

O grafo base dos grafos testados possuem 286 vértices distribuídos entre 25 clusters que contém de 9 a 15 vértices.

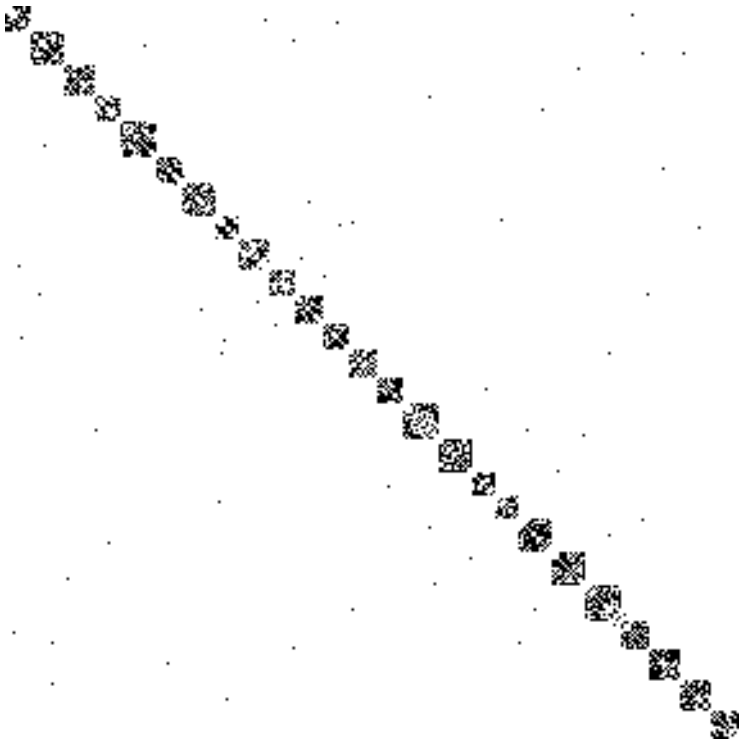
O primeiro dos grafos testados foi gerado com uma densidade interna de 0.6 e razão de 0.03 entre arestas inter e intra clusters, gerando um total de 941 arestas, dentre elas 913 são intra-clusters e apenas 28 inter-clusters. Podemos melhor visualizar o grafo através de sua matriz de adjacência representada pela figura 8.1.

Sabemos que cada α gera uma partição. No grafico representado abaixo, as medidas, que são representadas pelas retas, estão relacionadas as partições geradas por diferentes valores de α. Seja $P(\alpha)$ a partição gerada por um determinado α, então é calculado $\Psi(P(\alpha))$, $\Phi(P(\alpha))$, $\mu_\sigma(P(\alpha))$, a maior medida de corte do primeiro termo da expressão de α, e o próprio α para se ter um referencial.

Sabemos que o problema da condutância é $\mathbb{NP}$-completo. Se tivermos uma partição com clusters de tamanhos grandes, calcular o valor exato da medida de uma partição fica muito difícil. Assim, para o cálculo das medidas de condutância e expansão, são desconsiderados clusters com tamanho maior que 25, que ocorrem quando α é bem pequeno apenas. E o valor da medida para partições com tamanho maiores que 25 poderá ficar maior que o valor exato.

Para a análise do gráfico da figura 8.2 é desconsiderando as possíveis falhas dos cálculos das medidas de condutância e expansão. Existem dois fatos notórios que podem ser identificados

Figura 8.1: Matriz de adjacência de um grafo com 286 vértices e 941 arestas.



no gráfico acima. O primeiro deles é a medida de comparação com o ótimo, que em um dado momento atinge o valor 1 que indica que a aplicação foi capaz de reconhecer com exatidão a partição tida como ótima. O segundo deles é o α em relação as medidas de expansão e o corte. A todo momento α se encontra entre as duas medidas, e há um momento em que α se iguala a expansão.

As partições geradas com valores de α maior que a da partição ótima, contém muitos clusters unitários, que prejudicam as medidas, pois tanto a expansão como condutância resultam em infinito e a medida de corte em valor no máximo 1, pois o denominador dessa medida é alto para um cluster só.

Outro gráfico que se destaca é o gráfico que relaciona o primeiro e o terceiro termo da equação de α, o corte e a expansão, representado pela figura 8.3.

O gráfico representado por 8.3 é interessante, pois é desejável obter altos valores para a expansão, ao mesmo tempo que é desejável obter valores baixos para a medida de corte. Assim seria ideal obter pontos mais próximos do canto superior esquerdo do gráfico. O ponto que melhor traduz essa ideia é o ponto central do gráfico, que é exatamente a partição ótima.

O teste anterior foi bastante fácil, pois as arestas inter-clusters eram escassas. Agora fare-

Figura 8.2: α x medidas

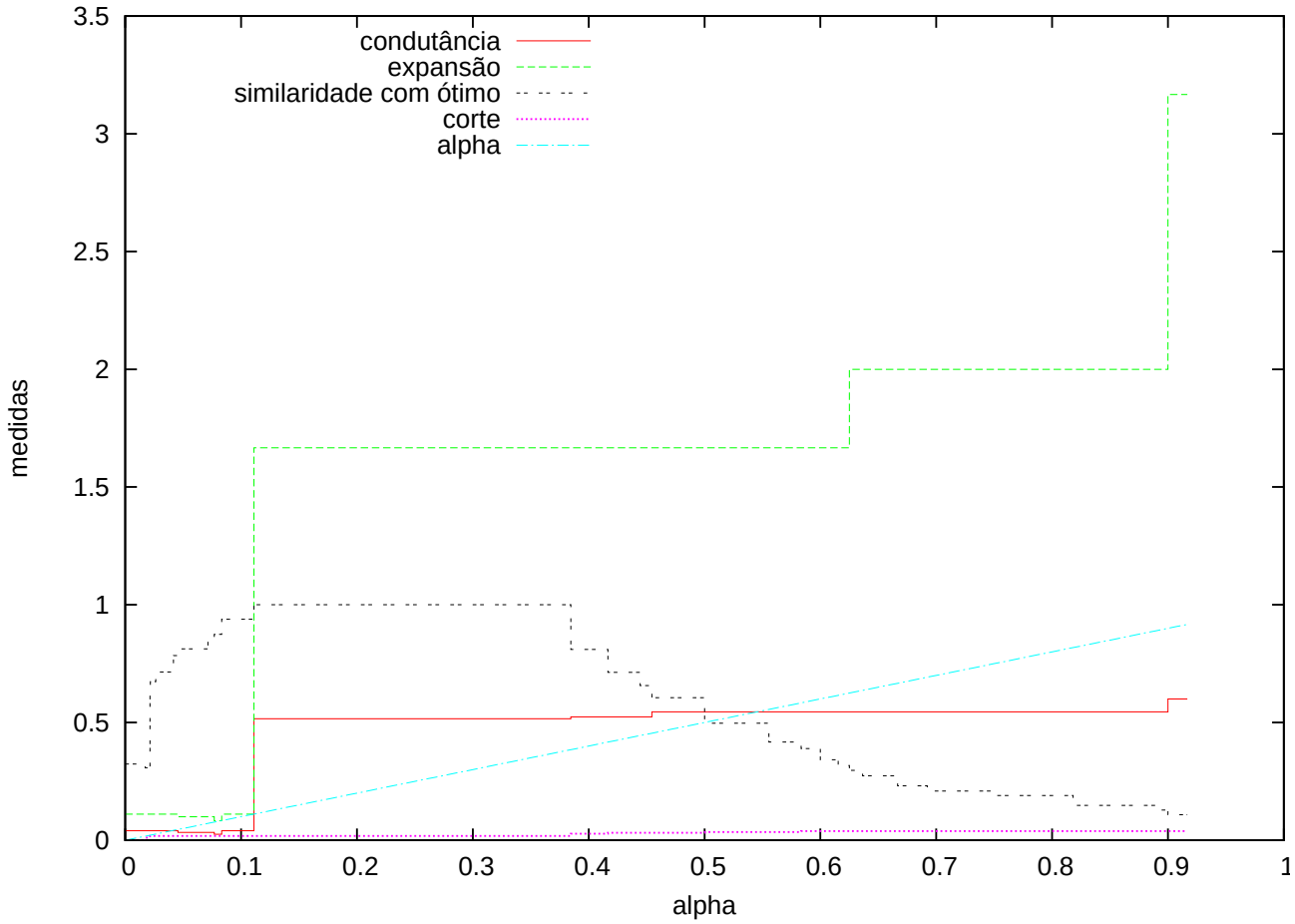
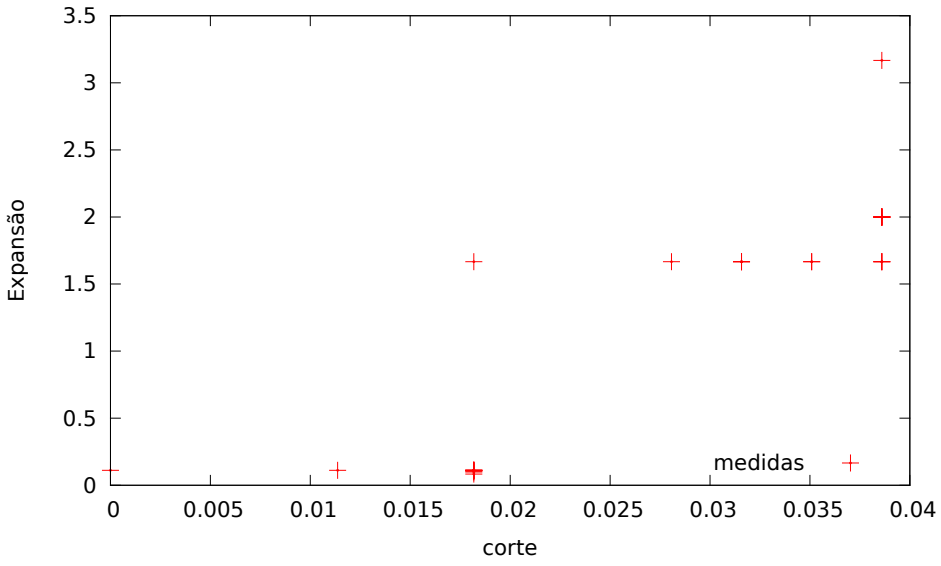
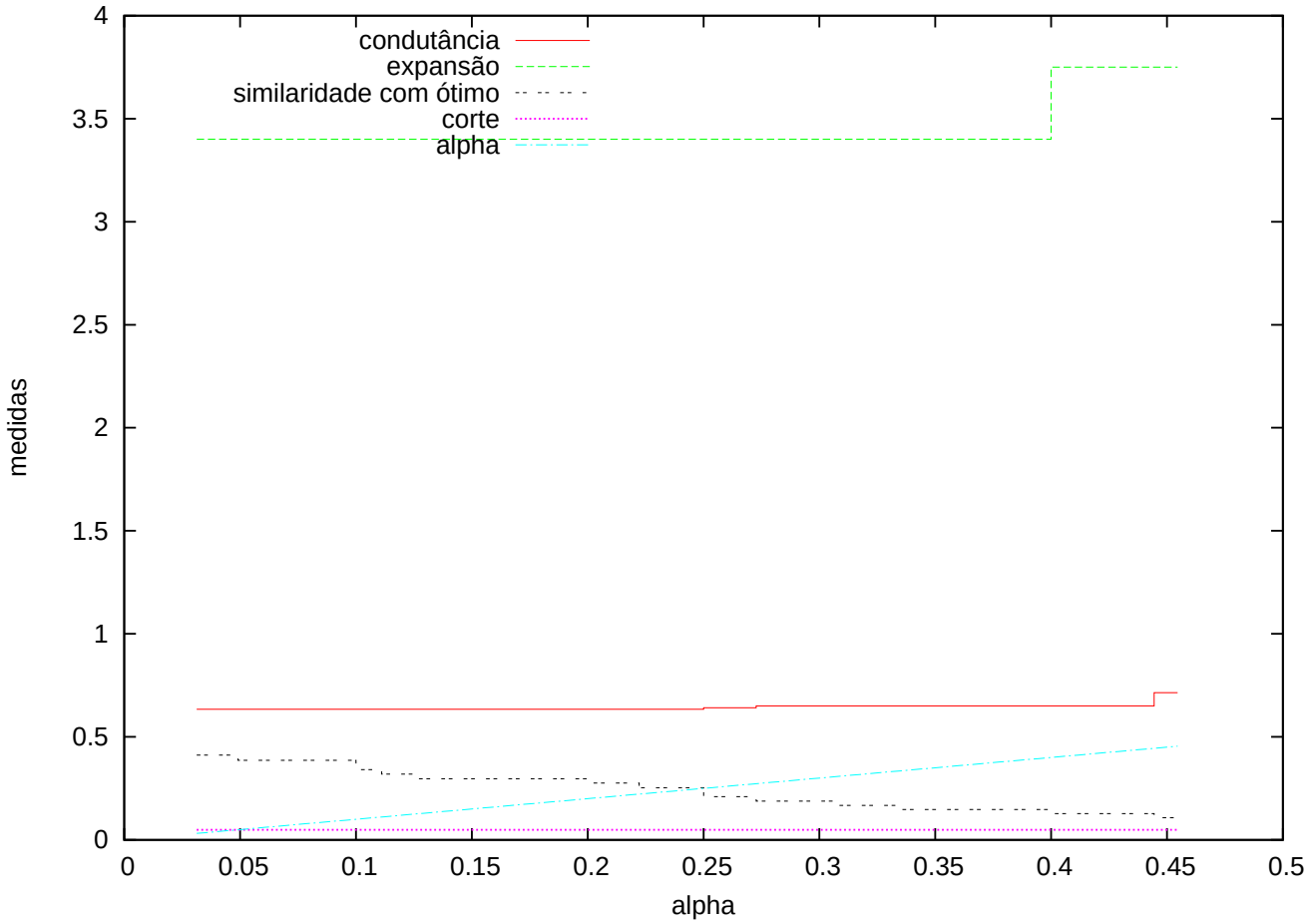


Figura 8.3: medida de corte *x* expansão



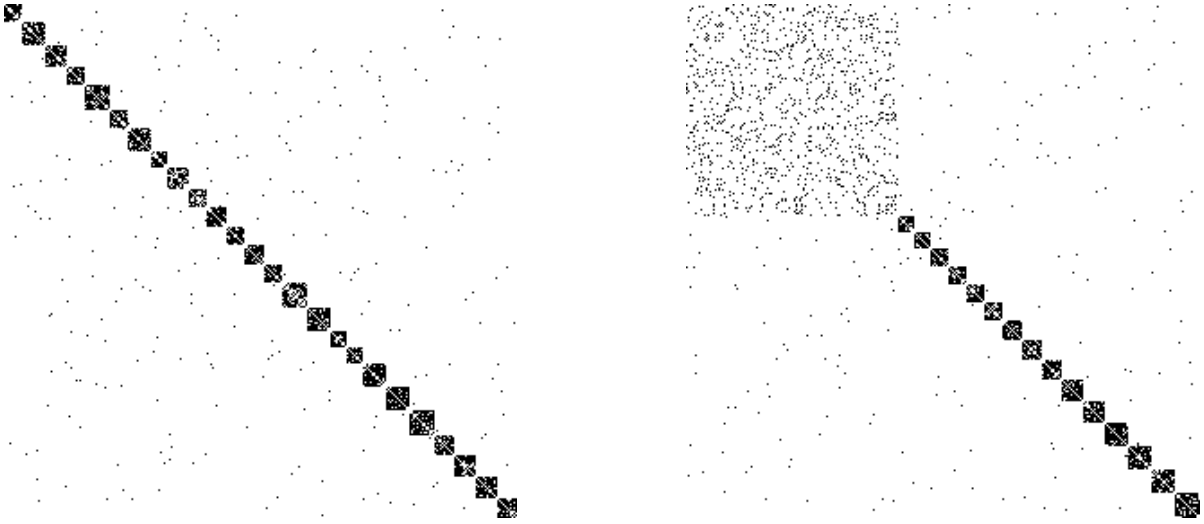
mos um teste um pouco mais complicado, com as arester inter-clusters representando mais das arestas, assim foi escolhido o valor 0.8 para a densidade interna e 0.1 a razão. Assim foram geradas 1338 arestas, das quais 1216 são intra e 122 inter.



Agora o gráfico é bem desinteressante, pois as partições geradas possui pouca similaridade com a partição ótima. Além disso, as outras medidas assumem valor praticamente constante, e o gráfico que relaciona corte e expansão se resumem a dois pontos em posições irrelevantes. Comparando este gráfico com o mesmo do outro grafo, parece que está faltando informação a esquerda. Ou seja a impressão é que era possível identificar outras partições com α menores, porém não foram identificados. Talvez o principal motivo disso está na equação do α, pois não existiram cortes para o primeiro termo da equação de α que fossem baixos o suficiente para a identificação de novas partições.

Porém, analisando a partição com maior similaridade com o grafo original, percebe-se que foi possível a identificação correta de alguns clusters, contudo os demais clusters identificados foram unitários. Apesar de tudo, conseguiu-se uma partição razoável como pode ser melhor identificada pelas matrizes de adjacência da figura 8.4.

Figura 8.4: matrizes de adjacências do mesmo grafo, com a partição original(esquerda) e a partição gerada pela aplicação(esquerda)



A primeira matriz de adjacência é o grafo otimamente particionado, enquanto que a segunda é a partição gerada pela aplicação. Os vértices de cada partição estão dispostos de uma forma tal que os vértices de um cluster fiquem adjacentes. A primeira nuvem notória na figura da direita foram todos os cluster unitários identificados, enquanto as outras nuvens mais densas de pontos são os clusters relevantes gerados.

8.2 Avaliação

Além dos dois grafos acima apresentados, foram feitos experimentos com outros tipos de grafos, como o *Relaxed Caveman Graph* [1], obtendo resultados semelhantes.

Também foi testados em grafos otimamente particionados com números elevados de arestas inter-clusters, porém a aplicação foi incapaz de obter bons resultados. As partições geradas para tais grafos possuiam muitos clusters unitários e poucos clusters relevantes. O principal motivo que tenha acontecido isso é similar com o motivo do segundo experimento, e está relacionado com o primeiro termo da equação do teorema principal de *CutClustering*.

Todos os grafos testados aqui foram grafos sem peso, e o algoritmo de *CutClustering* foi projetado para particionar também grafos com peso. Não foram gerados grafos com peso para testar em cima da aplicação neste trabalho, porém foi obtido alguns bons resultados em [2].

9 Conclusão

Neste trabalho foi estudado o problema de *Clustering*. Como o problema não possui uma definição clara, foi feita uma definição que melhor condizia com o escopo do trabalho. Definido o problema, foi estudado a parte principal da definição, as medidas de qualidade. As medidas de qualidade constituem a base do problema, e a criação de medidas que melhor traduzem o problema é uma tarefa difícil, pois medidas boas normalmente são difíceis computacionalmente de se calcular.

O algoritmo aqui apresentado funciona bem para algumas configurações de grafos, principalmente para grafos muito esparsos. Para outros tipos de configuração de grafos, o algoritmo adotado não se comportou de uma maneira adequada e não apresentou boas partições. Tal fato já era esperado pela equação de seu teorema principal, que se comporta como uma faca de dois gumes.

Foi também estudado a geração de grafos artificiais para a criação de bases de dados para *benchmark*. Como esse assunto é de alta importância, foi necessário a definição e construção precisa de tais grafos.

Com a base de dados criada, o algoritmo apresentado foi testado em cima dessa base, e informações sobre sua saída foi analisada, obtendo informações interessantes.

Pode-se constatar que o problema descrito é de fato muito difícil. Para tentar obter resultados aceitáveis foi necessário um amplo estudo e gerou resultados nem sempre muito satisfatórios. *Clustering*, é um tema bastante estudado com muitas áreas ainda para se explorarem.

Referências Bibliográficas

[1] Universite De, Satu Virtanen, Satu Virtanen, and Multiprint Oy. Properties of nonuniform random graph models. Technical report, Helsinki University of Technology, Laboratory for Theoretical Computer Science, 2003.

[2] Gary W. Flake, Robert E. Tarjan, and Kostas Tsioutsiouliklis. Graph Clustering and Minimum Cut Trees. *Internet Mathematics*, 1(4):385–408, 2004.

[3] Gary William Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of web communities. In *In Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 150–160. ACM Press, 2000.

[4] L. R. Ford and D. R. Fulkerson. *Flows in networks, by L.R. Ford, Jr. [and] D.R. Fulkerson*. Princeton University Press, Princeton, N.J.,, 1962.

[5] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, 18(1):30–55, February 1989.

[6] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[7] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51:497–515, May 2004.

[8] L. Moussiades and A. Vakali. Benchmark graphs for the evaluation of clustering algorithms. In *Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on*, pages 197–206, april 2009.

[9] Barna Saha and Pabitra Mitra. Dynamic algorithm for graph clustering using minimum cut tree. In *Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops*, pages 667–671, Washington, DC, USA, 2006. IEEE Computer Society.

[10] S. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, August 2007.

[11] Maria Scutellà. A note on the parametric maximum flow problem and some related reoptimization issues. *Annals of Operations Research*, 150:231–244, 2007. 10.1007/s10479-006-0155-z.

[12] Jiri Sima and Satu E. Schaeffer. On the NP-Completeness of Some Graph Cluster Measures. Jun 2005.