Clustering consensual para a detecção eficiente de comunidades em redes pela modularidade ajustada

Camila Pereira dos Santos[1], **Mariá Cristina Vasconcelos Nascimento**[1]

[1]Instituto de Ciência e Tecnologia - Universidade Federal de São Paulo - UNIFESP
Rua Talim, 330, CEP: 12231-280. Tel: +55 12 3309-9595/ +55 12 3921-5717

{santos.camila, mcv.nascimento}@unifesp.br

Abstract. *The modularity maximization is the most used approach to detect communities in networks. Nevertheless, it can fail to find small sized communities, due to a scale problem. In order to overcome this problem, an adjusted version of modularity was proposed in literature. Despite presenting good potential, studies that use this measure to find clusterings in an automatic fashion were not found. In this paper, it is proposed a method to automatically determine, by means of the concept of consensual clustering, clusterings by using the adjusted modularity. Computational experiments using several graphs attested that the proposed strategy outperformed several graph clustering algorithms from literature.*

Resumo. *A maximização da modularidade é a abordagem mais utilizada para detectar comunidades em redes. Contudo, ela pode falhar em encontrar comunidades pequenas, devido a um problema de escala. Para superar esse problema, uma versão ajustada da modularidade foi proposta na literatura. Apesar de seu potencial, não foram encontrados estudos que utilizassem essa medida para encontrar agrupamentos de maneira automática. Neste artigo, é proposto um método para determinar automaticamente comunidades por meio da modularidade ajustada, usando o conceito de clustering consensual. Experimentos com diversos grafos atestaram um melhor desempenho da estratégia proposta sobre diversos algoritmos de agrupamento em grafos da literatura.*

1. Introdução

Grafos constituem uma forma bastante utilizada para representar diversos sistemas reais como, por exemplo, redes sociais [Traud et al. 2008]. Muitas redes reais possuem uma propriedade conhecida como estrutura de comunidade [Girvan and Newman 2002] que pode ser entendida como a tendência de agrupamento dos vértices, ou seja, de existirem grupos de vértices altamente relacionados. Detectar comunidades em redes[1] consiste, portanto, em identificar essa estrutura, de modo a agrupar vértices fortemente conectados [Schaeffer 2007] e formar um *clustering*.

Métodos baseados na maximização da modularidade são os mais comumente utilizados [Girvan and Newman 2002, Aloise et al. 2012, Noack and Rotta 2009] para detecção de comunidades em redes. Entretanto, [Fortunato and Barthélemy 2007] apontam a existência de um limite de resolução na modularidade. Isso implica que comu-

[1]Agrupamento em grafos tem o mesmo significado de detectar comunidades em redes.

nidades menores que uma escala definida pelo tamanho do grafo e do número de comunidades presentes nele podem não ser devidamente encontradas. [Newman 2012] confirmam essa dependência, porém afirmam que o ajuste na modularidade, proposto por [Reichardt and Bornholdt 2006] e [Arenas et al. 2008], pode superar essa dependência. A medida com o parâmetro de ajuste é chamada, neste artigo, de *modularidade ajustada*.

O potencial de encontrar boas soluções, principalmente, para os casos nos quais a maximização da modularidade apresenta limite de resolução, por meio da modularidade ajustada, não tem sido explorado na literatura. Mais especificamente, não se conhece nenhuma maneira de ajustar automaticamente a modularidade ajustada, que é dependente de um parâmetro. Dessa maneira, neste artigo, um algoritmo que encontre um *clustering* consensual dentre as comunidades encontradas a partir de um intervalo de valores para o parâmetro da modularidade ajustada é proposto. Algoritmos de *clustering* consensual foram propostos na literatura por [Lancichinetti and Fortunato 2012] e por [Topchy et al. 2005]. Nos testes computacionais, é verificada a superioridade dos resultados obtidos pelo algoritmo proposto não só sobre métodos baseados na maximização da modularidade, mas também sobre outras estratégias bem conhecidas de detecção de comunidades em redes: *spinglass* [Reichardt and Bornholdt 2004, Reichardt and Bornholdt 2006] , *edge betweeness* [Newman and Girvan 2004] , *label propagation* [Raghavan et al. 2007] e *infomap* [Rosvall and Bergstrom 2008].

2. Caracterização do Problema

A medida conhecida por modularidade, proposta por [Girvan and Newman 2002] avalia a qualidade de um *clustering* obtido a partir de uma dada partição do conjunto de vértices de um grafo. Um grafo é definido por $G = (V, E)$, sendo $V(G)$ seu conjunto de vértices e $E(G)$ o seu conjunto de arestas. Neste artigo, elementos de $V(G)$ são representados por números naturais de 1 a $|V(G)|$, e elementos de $E(G)$ por tuplas (u, v) tais que u e $v \in V(G)$. Um *clustering* $\mathcal{C}$ é definido por $\mathcal{C} = \{C_1, C_2, ..., C_k\}$, em que $C_1, C_2, ..., C_k$ representam os seus *clusters*. Para avaliar a qualidade de um *clustering*, estima-se a quantidade total de arestas dentro dos *clusters*[2] em relação à quantidade esperada de arestas em um grafo aleatório com a mesma sequência de graus dos vértices do grafo estudado. Dessa maneira, quando houver mais arestas dentro dos *clusters* que o esperado em um grafo aleatório com a mesma sequência de vértices, é considerado que os *clusters* avaliados tenham uma tendência de agrupamento. Uma formulação para essa medida é apresentada na Equação (1).

$$q(\mathcal{C}) = \frac{1}{2m} \sum_{C \in \mathcal{C}} \sum_{i,j \in C} \left(a(i,j) - \frac{d_g(i) d_g(j)}{2m} \right) \tag{1}$$

Na Equação (1), m indica a quantidade total de arestas do grafo, $\mathcal{C}$ indica um *clustering*, $a(i, j)$ indica a presença ou ausência de uma aresta entre os vértices i e j e $d_g(i)$ indica o grau do vértice i, definido por $d_g(i) = \sum_{j=1}^{|V(G)|} a(i, j)$.

[Brandes et al. 2008] provam que os valores possíveis para a modularidade pertencem ao intervalo $[-\frac{1}{2}, 1]$. Valores mais altos, próximos ao valor máximo 1, indicam

[2]Este artigo utiliza a nomenclatura *cluster* com o mesmo sentido de comunidade.

clusterings de melhor qualidade. O problema da maximização da modularidade objetiva encontrar um *clustering* com modularidade máxima, ou seja, dentre o conjunto de todas as partições de um conjunto de vértices, aquela que possua a maior modularidade. Contudo, o problema de decidir se uma partição tem modularidade máxima é NP-Completo, de acordo com [Brandes et al. 2008]. Métodos de solução para resolver esse problema de forma aproximada encontrados na literatura são as heurísticas, como, por exemplo, metaheurísticas [Aloise et al. 2012, Arenas et al. 2008] e algoritmos espectrais [Newman 2006].

2.1. Limite de resolução

Apesar de uma boa fundamentação teórica, a medida de modularidade apresenta falhas. Segundo [Fortunato and Barthélemy 2007] a modularidade possui um limite de resolução, ou seja, podem ocorrer falhas em detectar comunidades menores que uma escala definida pelo tamanho do grafo e pelo grau de interconectividade entre pares de comunidades. [Fortunato and Barthélemy 2007] apresentam, ainda, casos em que subgrafos de um grafo, conectados por apenas uma aresta, podem não ser identificados como *clusters* se não respeitarem o limite de resolução. Ainda segundo [Fortunato and Barthélemy 2007], ao ignorar comunidades pequenas, a medida de modularidade é tendencionada a encontrar comunidades grandes.

[Reichardt and Bornholdt 2006] mostram que o problema de encontrar o estado fundamental de um vidro de *spin* com alcance infinito pode ser utilizado para mapear o problema da detecção de comunidades em redes. Os autores concluem que existe uma relação entre esses dois problemas, considerando o aparecimento de um parâmetro λ que ajusta a escala que a modularidade avalia as comunidades. Essa medida, modularidade ajustada, é apresentada na Equação (2). Nela, λ indica o parâmetro que controla o ajuste da modularidade, que quando é 1, nada mais é do que a modularidade em sua forma original.

$$Q(\mathcal{C}) = \frac{1}{2m} \sum_{C \in \mathcal{C}} \sum_{i,j \in C} \left(a(i,j) - \lambda \frac{d_g(i) d_g(j)}{2m} \right) \tag{2}$$

Encontrar uma partição que maximize a modularidade ajustada requer a escolha de um valor apropriado para o parâmetro λ. Este artigo propõe que sejam obtidas partições usando diversos valores de λ para que, em seguida, seja obtido um *clustering* consensual das mesmas. De acordo com [Topchy et al. 2005], a ideia do *clustering* consensual é combinar diferentes partições, de forma a obter um *clustering* final de boa qualidade. [Lancichinetti and Fortunato 2012] aplicam uma estratégia de clustering consensual em diversos algoritmos da literatura e concluem que essa estratégia melhora os resultados obtidos pelos algoritmos. Mais detalhes da estratégia para obtenção do *clustering* consensual e do algoritmo proposto neste artigo são apresentados na seção seguinte.

3. Estratégia Multi-nível adaptada para GRASP

Estratégias multi-níveis são métodos para encontrar partições em grafos de forma eficiente compostos por três fases: de contração, na qual os vértices são sucessivamente contraídos, de particionamento, na qual o *clustering* é realizado, e de refinamento, na

qual os vértices contraídos são expandidos e buscas locais são realizadas. Em particular, [Noack and Rotta 2009] propuseram algoritmos multi-níveis para o problema da maximização de modularidade.

Neste artigo, é proposto um algoritmo multi-nível baseado em uma metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP) [Feo and Resende 1995], aqui denominado GRASPML, para resolver o problema de maximização da modularidade ajustada. O GRASP é um método no qual cada iteração é composta por duas fases: a construtiva e a de busca local. A fase construtiva é caracterizada por sua aleatoriedade controlada, no qual adapta-se um algoritmo guloso para gerar soluções pseudo-aleatórias. Para isso, ao invés de construir a solução escolhendo dentre uma lista de melhores candidatos, a melhor opção, considera-se uma lista das melhores opções de inserção na solução. Após a construção de uma solução aplica-se a ela uma busca local, que consiste em, a partir da solução construída, encontrar uma solução melhor no espaço de busca.

A primeira fase do GRASPML consiste em uma modificação da estratégia gulosa de contração do algoritmo em [Noack and Rotta 2009] para uma estratégia semi-gulosa. A busca local da GRASP será realizada durante a fase de refinamento do *clustering* da estratégia multi-nível. No intuito de diminuir tempo de convergência da heurística, assim como oferecer um mecanismo de memória para o GRASP, pares de vértices que estejam frequentemente no mesmo *cluster* nos *clusterings* de um conjunto elite de soluções são contraídos. Isso permite que o tamanho do grafo seja reduzido, diminuindo o custo computacional da fase construtiva. O Algoritmo 1 apresenta o pseudocódigo do GRASPML.

Algoritmo 1: GRASPML

Entrada: grafo $G = (V, E)$, *maxIter* , *iterContr*, λ
Resultado: *clustering* $\mathcal{C}*$

```
1  G := preProcessamento(G)
2  para iter=1 até maxIter faça
3  |   G_k := faseContracao(G)
4  |   C := faseRefinamento(G_k)
5  |   atualizaMelhorSolucao(C*, C)
6  |   atualizaElite(E, C)
7  |   se iter > iterContr então
8  |   |   G := contracaoDefinitiva(G , E)
9  |   fim
10 fim
11 retorna C*
```

As entradas do Algoritmo 1 são o grafo G, o número máximo de iterações da GRASP, $maxIter$, a iteração que as contrações definitivas são efetuadas, $iterContr$, e um valor para o parâmetro λ da modularidade ajustada. Na linha 1, algumas situações especiais, como vértices desconexos no grafo, entre outras, são tratados pela função *preProcessamento(G)*. Na linha 3, a fase de contração é aplicada ao grafo G. Nela, sucessivas contrações de vértices são efetuadas, gerando uma sequência $(G_0, G_1, G_2, ..., G_k)$ com um número decrescente de vértices e arestas. Partindo do grafo G_0, que é o grafo original, é obtido o grafo G_k [Blum et al. 2011]. Para determinar quais vértices serão

contraídos, a *faseContracao(G)* parte de uma lista de pares de vértices, ordenados de acordo com o maior ganho de modularidade. A partir dessa lista, o algoritmo semiguloso de contração determina os pares de vértices que serão contraídos. Ao fim da contração, cada um dos vértices do grafo G_k é considerado um *cluster*. Na linha 4, a *faseRefinamento(G_k)* é responsável por, a partir de G_k, realizar a expansão do grafo atribuindo a cada um dos vértices de G_{k-1} o *cluster* do supernó[3] no qual foram contraídos em G_k [Dhillon et al. 2005]. Após cada expansão, é aplicada uma busca local no *clustering*, a fim de melhorar a sua qualidade. Na busca local procura-se repetidademente pelo melhor movimento de transferência de vértices de um *cluster* para outro. Esses movimentos são realizados até que não seja possível nenhuma melhora na modularidade e o ótimo local seja garantido. Na linha 5, a melhor solução $\mathcal{C}*$ é atualizada, por meio da função *atualizaMelhorSolucao($\mathcal{C}, \mathcal{C}*$)*, caso $Q(\mathcal{C}) > Q(\mathcal{C}*)$. Na linha 6, a função *atualizaElite($\mathcal{E}, \mathcal{C}$)* atualiza o conjunto de soluções de elite $\mathcal{E}$, com o clustering $\mathcal{C}$ caso seja necessário. Na linha 8, caso já tenham sido executadas mais de $iterContr$ iterações, é executada a função *contracaoDefinitiva($G, \mathcal{E}$)*. Nela, são contraídos os pares de vértices que estejam no mesmo *cluster* em todos os *clusterings* de $\mathcal{E}$, com uma probabilidade de 50%. A solução retornada $\mathcal{C}*$ é o *clustering* $\mathcal{C}$ com o maior valor $Q(\mathcal{C})$ encontrado.

A fim de validar a GRASPML como um método eficiente para o problema maximização da modularidade, foram realizados experimentos com diversos grafos clássicos da literatura, comparando a modularidade obtida pelo GRASPML em relação ao estado da arte. A GRASPML encontrou valores de modularidade próximos aos melhores reportados na literatura, apesar de não superá-los [4].

4. CGRASPML

Para definir o *clustering* final de maneira automática, apenas fornecendo um conjunto finito de valores de λ para o GRASPML, utilizou-se a ideia de *clustering* consensual [Topchy et al. 2005]. Um *clustering* consensual consiste em um *clustering* obtido a partir da análise de componentes em comum de um conjunto de *clusterings*. É com essa filosofia que se propõe a estratégia para determinação de um *clustering* consensual neste artigo.

Para isso, foram obtidos *clusterings* para a maximização da modularidade ajustada, utilizando o GRASPML, com λ variando de $0,1$ a 3, em intervalos de $0,1$. Esse intervalo foi estabelecido após um estudo teórico que indicou que esses valores abrangem a maior parte dos tipos dos grafos, independente de seu tamanho ou escala. Um *clustering* consensual foi obtido ao colocar no mesmo *cluster* os vértices que apareciam no mesmo *cluster* em mais de 50% das partições por todos os λ do intervalo.

5. Experimentos Computacionais

Esta seção apresenta três experimentos realizados para a validação da estratégia proposta, o CGRASPML. No experimento I, um grafo *benchmark* conhecido por *afootball* é analisado segundo resultados obtidos pelo CGRASPML. No experimento II, um grafo em forma de anel, para o qual a maximização da modularidade apresenta limite

[3]Um conjunto de vértices contraídos para formar outro vértice, no nível seguinte de contração do grafo, é chamado de supernó [Dhillon et al. 2005].

[4]Detalhes a respeito dessa experimentação, assim como dos resultados obtidos, podem ser encontrados no sítio: `https://sites.google.com/site/nascimentomcv/camila_santos`

de resolução, segundo [Fortunato and Barthélemy 2007], é analisado. No experimento III, utiliza-se um conjunto de grafos caracterizados por terem pequenos *clusters*, gerados por meio do software de [Lancichinetti et al. 2008], por [Nascimento 2013][5]. Nos experimentos I, II e III, os resultados obtidos pelo CGRASPML são comparados com os algoritmos clássicos de detecção de comunidades em redes: um muti-nível baseado na maximização da modularidade, aqui chamado de *Blondel* [Blondel et al. 2008] , *spinglass* [Reichardt and Bornholdt 2004, Reichardt and Bornholdt 2006], *label propagation* (LP) [Raghavan et al. 2007], *edge betweeness* [Newman and Girvan 2004] e *infomap* [Rosvall and Bergstrom 2008]. De acordo com os experimentos realizados por [Lancichinetti and Fortunato 2009] com diversos algoritmos de detecção de comunidades em redes, o *infomap* foi o que obteve os melhores resultados.

Adicionalmente, utiliza-se o GRASPML com $\lambda = 1$, que aqui é chamado apenas de GRASPML. Em ambos GRASPML e CGRASPML, os valores de seus parâmetros são: $maxIters = 30$ e $iterContr = 15$. Para comparar resultados entre *clusterings*, uma medida conhecida por NMI (*Normalized Mutual Information*) [Danon et al. 2005], que avalia a proximidade entre dois *clusterings*, será utilizada. Os valores do NMI vão de 0 a 1 e quanto maiores, mais parecidos são os *clusterings* avaliados. Todos os experimentos foram realizados em um computador com processador Intel Core i7 2600 3.4 GHz, com 16 GB de memória principal e com sistema operacional Windows 7.

5.1. Experimento I

O grafo clássico *afootball*, compilado por [Girvan and Newman 2002], possui vértices representando times de futebol americano. Arestas entre esses vértices existem se os pares de vértices jogaram repetidamente na temporada do ano 2000. Segundo [Girvan and Newman 2002], esse grafo é caracterizado por possuir 12 grupos de times, que os autores chamam de conferências, bem balanceados. Por ser um grafo com 115 vértices e 12 comunidades, esse grafo pode ser enquadrado na classe de grafos com comunidades pequenas. O resultado obtido por meio do CGRASPML aplicado a esse grafo é apresentado na Figura 1. Nessa figura, os rótulos numéricos indicam os *clusters* do grafo.

O NMI encontrado por meio do *clustering* encontrado por CGRASPML e o *clustering* esperado, que consiste nas conferências dos times, foi de 0,92. O algoritmo *infomap* também obteve um valor de 0,92 para o NMI. Dentre os demais algoritmos, o maior NMI foi encontrado pelo algoritmo *spinglass*, com um valor de 0,90. Em particular, o NMI encontrado por GRASPML foi de 0.86. Esse resultado indica um bom desempenho do CGRASPML para um grafo *benchmark* amplamente usado como referência de redes de relacionamento.

5.2. Experimento II

O grafo considerado neste experimento consiste de 30 cliques de 5 vértices cada, totalizando 150 vértices. Um clique é definido por um grafo simples no qual seus vértices possuem arestas para todos outros vértices do grafo. No caso de um clique de 5 vértices, seu número total de arestas é 10. Logo, nesse estudo de caso com 30 cliques, haverão 300

[5]Detalhes a respeito desses grafos podem ser encontrados no sítio: `https://sites.google.com/site/nascimentomcv/downloads/grafos`.

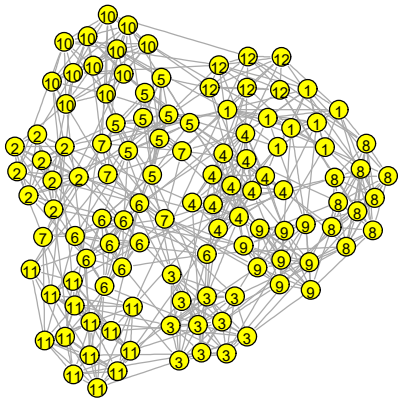

Figure 1. Grafo *afootball* com vértices rotulados segundo o algoritmo CGRASPML.

arestas totais internas aos cliques. Além disso, neste estudo de caso, cada um dos cliques é conectado a outro clique por apenas uma aresta. Portanto, haverá um total de 30 arestas externas aos cliques. Esse grafo está ilustrado neste artigo na Figura 2.

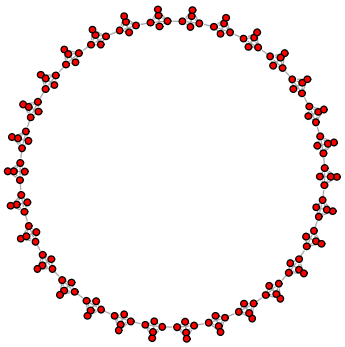

Figure 2. Grafo com 30 cliques de 5 vértices cada.

O *clustering* esperado para esse grafo é aquele no qual cada clique corresponde a um *cluster*, totalizando 30 comunidades no *clustering*. De acordo com [Fortunato and Barthélemy 2007], a solução ótima para a maximização da modularidade é um *clustering* de 15 comunidades, em que pares de cliques pertencem a uma mesma comunidade. Esse resultado é consequência do já mencionado limite de resolução da medida.

Neste experimento, o CGRASPML foi aplicado ao grafo em questão e, como resultado, obteve sucesso ao classificar cada clique em um *cluster*. Portanto, o NMI entre o *clustering* esperado e o *clustering* encontrado pelo CGRASPML é, obviamente, 1. Dentre os algoritmos considerados na experimentação, o *label propagation* e o *infomap* também obtiveram sucesso. Como esperado, o GRASPML com o valor 1 para o parâmetro λ, que caracteriza a modularidade pura, encontrou a solução ótima para o problema de maximização da modularidade que é o *clustering* de 15 *clusters*. Esse resultado aponta um bom indício do *CGRASPML* que encontrou o *clustering* desejado por meio de sua estratégia consensual.

5.3. Experimento III

Neste experimento, um conjunto maior de grafos foi usado para validação dos resultados do CGRASPML. Um total de 80 grafos foram gerados artificialmente com um coeficiente de mistura μ_t variando de $0,1$ até $0,8$, sendo 10 grafos para cada índice de mistura. O coeficiente de mistura indica o quão definidas estão as comunidades de um dado grafo. Para valores menores de μ_t, as comunidades podem ser ditas bem definidas, e para valores μ_t maiores, elas são mais dispersas no grafo. Os grafos gerados tem a tendência de formar comunidades pequenas, que é um caso especial no qual algoritmos baseados na maximização da modularidade tendem a falhar. A Figura 3 apresenta um gráfico com os valores de NMI médio dos resultados obtidos pelos sete algoritmos de detecção de comunidades em redes utilizados neste experimento. Essa média é em função de houverem 10 grafos diferentes para cada μ_t. A Figura 4 apresenta um gráfico com os tempos de execução, em segundos, dos algoritmos GRASPML e CGRASPML, propostos nesse artigo. Os resultados do GRASPML e do CGRASPML correspondem a média de 3 execuções independentes dos algoritmos.

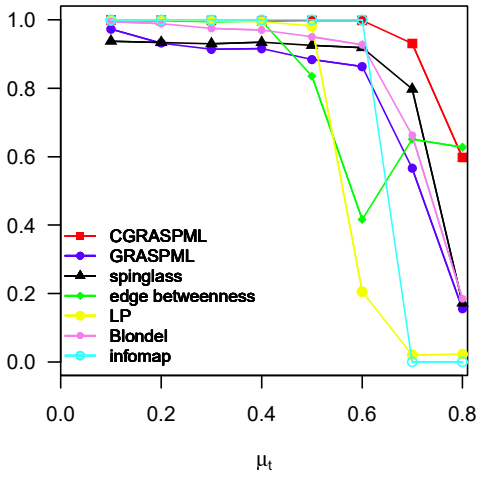

Figure 3. NMI médios para cada conjunto de grafos.

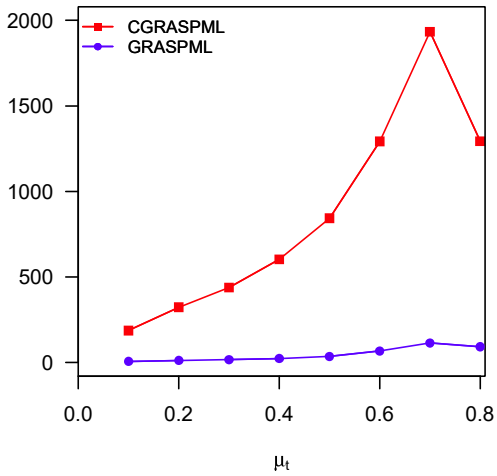

Figure 4. Tempos de execução do GRASPML e o CGRASPML.

A análise do gráfico apresentado na Figura 3 mostra que o CGRASPML obteve valores de NMI próximos a 1 para os grafos com μ_t até $0,6$. A partir desse μ_t, os valores de NMI começam a diminuir, apresentando uma queda brusca de $\mu_t = 0,7$ para $\mu_t = 0,8$. O CGRASPML obteve maiores valores de NMI que todos os outros métodos considerados, exceto para $\mu_t = 0,8$, que obteve um valor de NMI ligeiramente inferior ao obtido pelo *edge betweenness*. É importante considerar o comportamento imprevisível do *edge betweeness* que, ao contrário dos outros métodos, tem um pico de NMI com valor baixo em $\mu_t = 0,6$. Nesse experimento, observa-se que algoritmo *infomap* obteve valor 0 de NMI para os grafos com $\mu_t = 0,7$ e $\mu_t = 0,8$.

Em particular, o CGRASPML obteve maiores valores de NMI do que o GRASPML e o *Blondel*. Como o CGRASPML obteve um *clustering* a partir da modularidade ajustada e os outros dois a partir da modularidade pura, esses resultados são um indicativo de que a modularidade ajustada detecta melhor a estrutura de comunidade de grafos caracterizados por pequenas comunidades. Com isso, pode-se inferir que obter um *clustering* consensual por meio de *clusterings* gerados a partir da maximização da

modularidade ajustada com diversos valores de ajuste supera, ao menos em parte, o limite de resolução da modularidade. Em relação ao custo computacional do CGRASPML, observa-se, de acordo com a Figura 4, que é consideravalmente maior do que o custo computacional do GRASPML.

6. Considerações Finais

Este artigo apresenta uma maneira eficiente de se considerar uma medida de detecção de comunidades em redes que, até o momento, tem sido pouco utilizada apesar de seus indícios de bom potencial. Essa medida se trata de uma variação da modularidade [Girvan and Newman 2002], que, na sua forma original, é uma medida amplamente utilizada para encontrar comunidades. [Reichardt and Bornholdt 2006] propuseram um ajuste para a modularidade devido a um evento indesejável que afeta a medida sem ajuste, que se trata de comunidades pequenas não serem devidamente avaliadas. Esse erro de escala é conhecido por limite de resolução [Fortunato and Barthélemy 2007].

Neste artigo, uma estratégia para encontrar *clusterings* consensuais, chamada aqui de CGRASPML, por meio da maximização da modularidade ajustada, foi proposta. Para avaliar seus resultados, algoritmos para a detecção de comunidades em redes além dos baseados na maximização da modularidade foram usados neste experimento. Como consequência, o CGRASPML apresentou resultados significativamente melhores do que os demais algoritmos, considerando um grafo *benchmark* de redes sociais, um grafo conhecido por consistir de um caso no qual a modularidade apresenta limite de resolução, além de 80 grafos gerados pelo *software* de [Lancichinetti et al. 2008]. Entretanto, o CGRASPML é bastante custoso computacionalmente, de forma que trabalhos futuros devem propor alterações que diminuam seu custo computacional.

Agradecimentos

As autoras deste artigo agradecem a FAPESP pelo suporte financeiro.

Referências

Aloise, D., Caporossi, G., Hansen, P., Peron, S.and Liberti, L., and Ruiz, M. (2012). Modularity maximization in networks by variable neighborhood search. *Proceeding of 10h DIMACS Implementation Challenge*.

Arenas, A., Fernández, A., and Gómez, S. (2008). Analysis of the structure of complex networks at different resolution levels. *New J. Phys.*, 10(0530039).

Blondel, V. D., loup Guillaume, J., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*, (10).

Blum, C., Puchinger, J., Raidl, G. R., and Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11:4135–4151.

Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikolosk, Z., and Wagner, D. (2008). On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20:172–188.

Danon, L., Díaz-Guilera, A., Duch, J., and Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008–[11 pages].

Dhillon, I., Guan, Y., and Kulis, B. (2005). A fast kernel-based multilevel algorithm for graph clustering. *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 629–634.

Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Fortunato, S. and Barthélemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36.

Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99:7821–7826.

Lancichinetti, A. and Fortunato, S. (2009). Community detection algorithms: A comparative analysis. *Phys. Rev. E*, 80:056117.

Lancichinetti, A. and Fortunato, S. (2012). Consensus clustering in complex networks. *Scientific Reports*, 2.

Lancichinetti, A., Fortunato, S., and F, R. (2008). Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78:046110–[5 pages].

Nascimento, M. (2013). Community detection in networks via a spectral heuristic based on the clustering coefficient. *Submetido ao Discrete and Applied Mathematics*.

Newman, M. (2012). Communities, modules and large-scale structure in networks. *Nature Physics*, 8:25–31.

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

Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review*, E 69(026113).

Noack, A. and Rotta, R. (2009). Multi-level algorithms for modularity clustering. *SEA'09 Proceedings of the 8th International Symposium on Experimental Algorithms*, 1.

Raghavan, U. N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3 Pt 2):036106.

Reichardt, J. and Bornholdt, S. (2004). Detecting fuzzy community structures in complex networks with a potts model. *Phys. Rev. Lett.*, 93(21):218701.

Reichardt, J. and Bornholdt, S. (2006). Statistical mechanics of community detection. *Physical Review E*, 74:016110.

Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.

Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, 1:27–64.

Topchy, A., Jain, A., and Punch, W. (2005). Clustering ensembles: models of consensus and weak partitions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(12):1866–1881.

Traud, A. L., Kelsic, E. D., Mucha, P. J., and Porter, M. A. (2008). Community structure in online collegiate social networks. *arXiv:0809.0960*.