

Redes Complexas

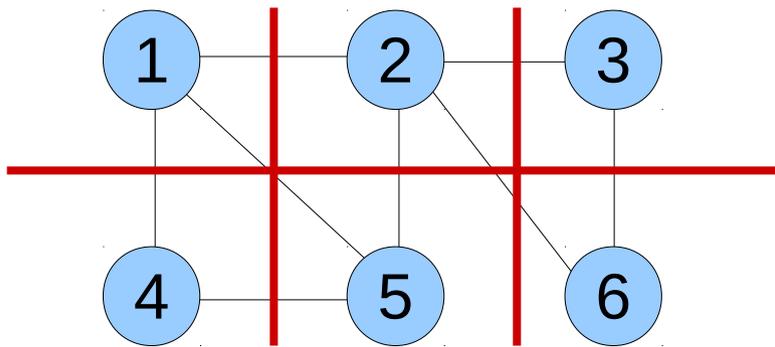
Aula 14

Roteiro

- Partição e bisseção em grafos
- Modularidade
- Algoritmos de Newman e Louvain
- Limitações

Partição de Grafos

- **Problema:** Dividir um grafo em partes
 - partição do conjunto de vértices



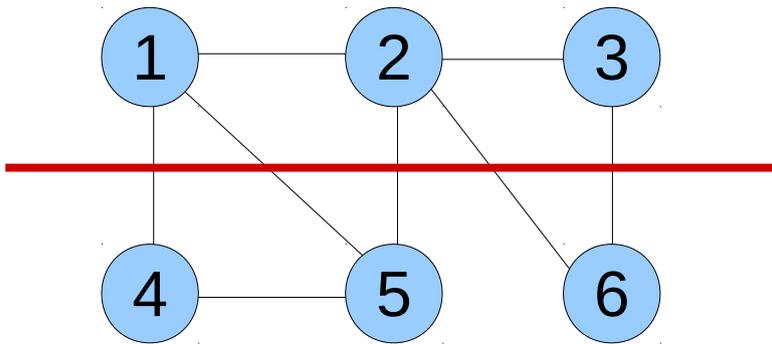
- Problema antigo, difícil e com muitas aplicações!
- Qual partição ao lado é melhor?

Como definir uma boa partição?

- Não há bala de prata, depende da aplicação

Corte em Grafos

- **corte:** induz uma partição com duas partes
 - subconjuntos A e B de vértices

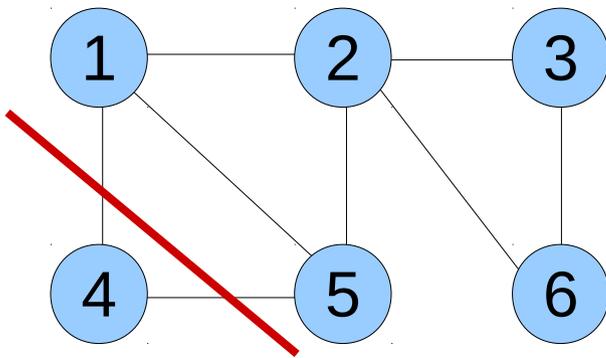


- $A = \{1, 2, 3\}$, $B = \{4, 5, 6\}$

- $c(A, B) =$ custo do corte $(A, B) =$ número de arestas entre vértices de A e B
 - soma dos pesos das arestas, em caso com peso
 - ex. $c(A, B) = 5$

Corte em Grafos

- **Problema:** encontrar o corte de custo mínimo de um grafo qualquer
 - problema clássico da computação, temos algoritmo polinomial

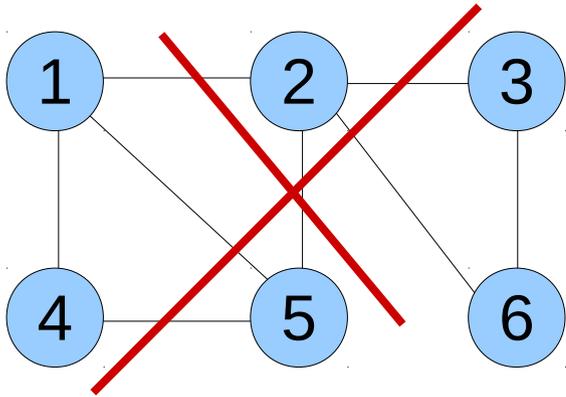


- $A = \{1, 2, 3, 5, 6\}$, $B = \{4\}$
- $c(A, B) = 2$

- **Problema:** tamanho das duas partes encontradas não é considerado
 - em muitas aplicações, isto é importante

Bisseção em Grafos

- **bisseção:** corte com duas partes do mesmo tamanho (ou 1 de diferença)
 - restrição na definição de corte



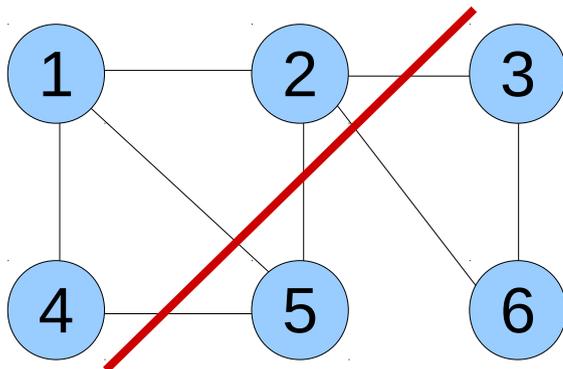
- $A = \{1, 2, 4\}$, $B = \{3, 5, 6\}$

- $c(A, B) = 5$

- **Problema:** encontrar a bisseção de custo mínimo em um grafo qualquer
 - ex. $A = \{1, 4, 5\}$, $B = \{2, 3, 6\}$, $c(A, B) = 2$

Bisseção em Grafos

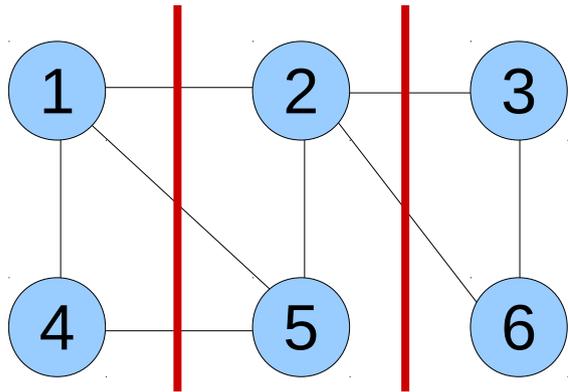
- Problema de bisseção é NP-Completo (não temos algoritmo polinomial)
 - exceções para classes de grafos, ex. árvores
- Algoritmo de Kernighan–Lin (1970)
 - algoritmo guloso (heurística) que troca pares de vértices até não reduzir mais o custo
 - a cada passo, escolhe par que ao trocar de lado mais reduz custo do corte



- $A = \{1, 2, 4\}$, $B = \{3, 5, 6\}$, $c(A, B) = 5$
- Trocar 2 e 5
- $A = \{1, 4, 5\}$, $B = \{2, 3, 6\}$, $c(A, B) = 2$

Partição Balanceada em Grafos

- Generalização da bisseção em grafos
- k partes de tamanho igual (ou 1 de diferença)
 - V_1, V_2, \dots, V_k , interseção é vazia, união é V
- Custo da partição é o custo do corte somado para todas as partes (V_i, V_j)

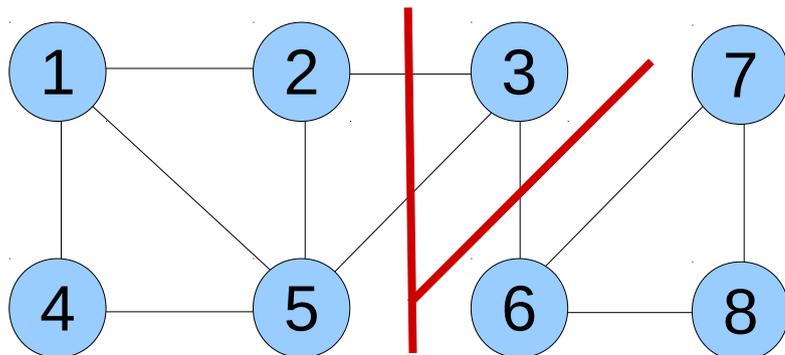


- $k=3, V_1 = \{1, 4\}, V_2 = \{2, 5\}, V_3 = \{3, 6\}$

- $\text{Custo} = c(V_1, V_2) + c(V_1, V_3) + c(V_2, V_3) = 3 + 0 + 2 = 5$

Problemas com Partição Balanceada

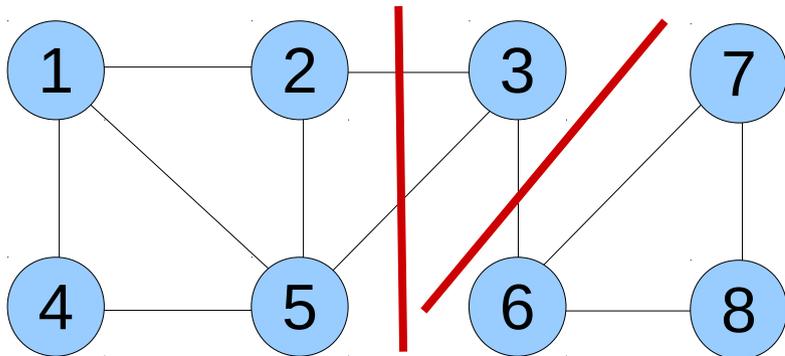
- **Problema 1:** número de partes tem que ser especificado
- **Problema 2:** tamanho das partes tem que ser igual (ou diferir em um)
- **Problema 3:** estrutura interna das partes não é considerado
 - podem não ser conexas, ter baixa densidade



- Em muitas aplicações, isto é importante
- *Network science* x teoria dos grafos!

Medindo Qualidade de Partição

- Como melhor aferir a qualidade de uma partição?
- **Ideia:** usar tamanho das partes para determinar “potencial” de arestas
 - normalização
- $\text{Ratio-Cut}(A,B) = c(A,B) / |A||B|$
 - onde $|A|$ é o número de vértices em A



- $\text{rc}(A,B) = 2 / (4*4) = 1/8$

- $\text{rc}(A,B) = 1 / (5*3) = 1/15$

Partição com Ratio Cut

- **Problema:** encontrar partição (com duas partes) que minimiza ratio cut
 - problema é NP-Completo
 - algoritmo aproximativo de Arora, Rao & Vazirani (2009), com fator $O(\sqrt{\log n})$
- Ratio cut tem conexões com decomposição spectral do Laplaciano do grafo
 - método para encontrar corte, limitante superior para ratio cut mínimo
- Ratio Cut ainda não resolve problemas
 - número de partes da partição
 - estrutura interna das partes

Modularidade

- **Outra ideia:** considerar modelo aleatório para criação de arestas, aferir arestas internas às partes

- p_{ij} : probabilidade de aresta entre vértices i e j

- Considere parte V_k (conjunto de vértices)

$$X_k = \frac{1}{2} \sum_{i, j \in V_k} A_{ij} = \text{número de arestas dentro de } V_k$$

$$Y_k = \frac{1}{2} \sum_{i, j \in V_k} p_{ij} = \text{número médio de arestas dentro de } V_k$$

$X_k - Y_k \rightarrow$ indica a “força” das arestas dentro de V_k

- Considerando todas as partes

$$M = \frac{1}{m} \sum_k (X_k - Y_k) = \frac{1}{2m} \sum_k \sum_{i, j \in V_k} (A_{ij} - p_{ij})$$

← constante de normalização (número de arestas)

Modularidade

■ **Problema:** Como definir p_{ij} para uma rede qualquer?

■ rede real não sai de um modelo

■ **Ideia:** usar o configuration model

■ assumir que rede foi gerada pelo configuration model

$$p_{ij} = \frac{d_i d_j}{2m} \rightarrow d_i, d_j \text{ são os graus dos vértices } i, j$$

■ Substituindo, temos

$$M = \frac{1}{2m} \sum_k \sum_{i, j \in V_k} \left(A_{ij} - \frac{d_i d_j}{2m} \right)$$

Modularidade

- Simplificando, temos

$$M = \sum_k \left(\frac{m_k}{m} - \left(\frac{d_k}{2m} \right)^2 \right)$$

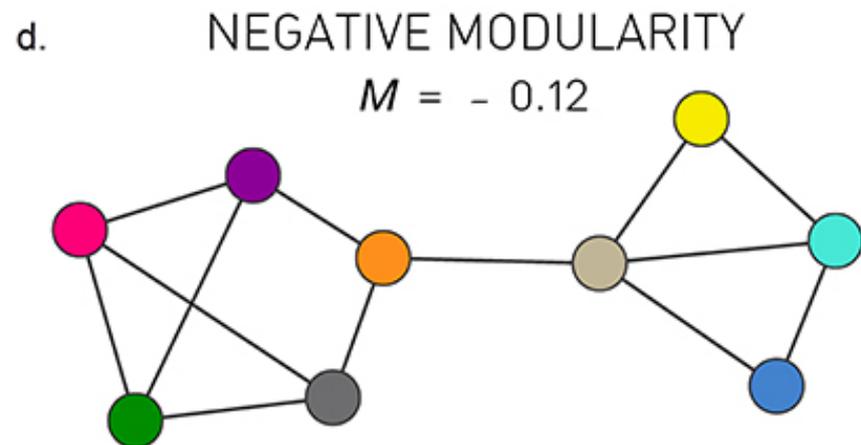
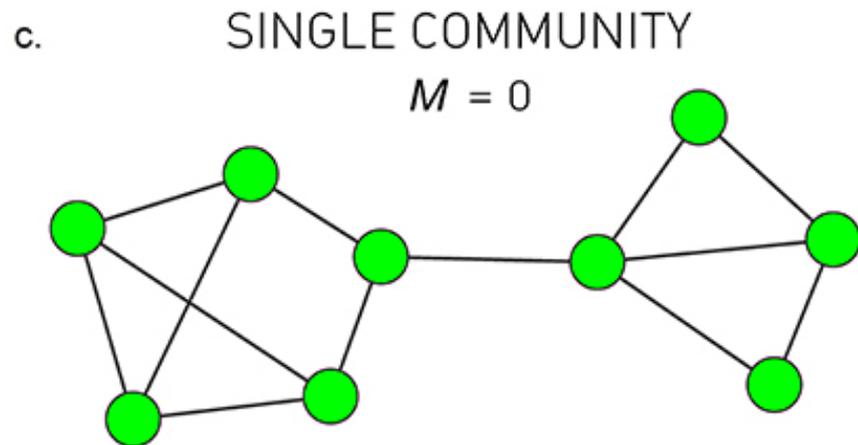
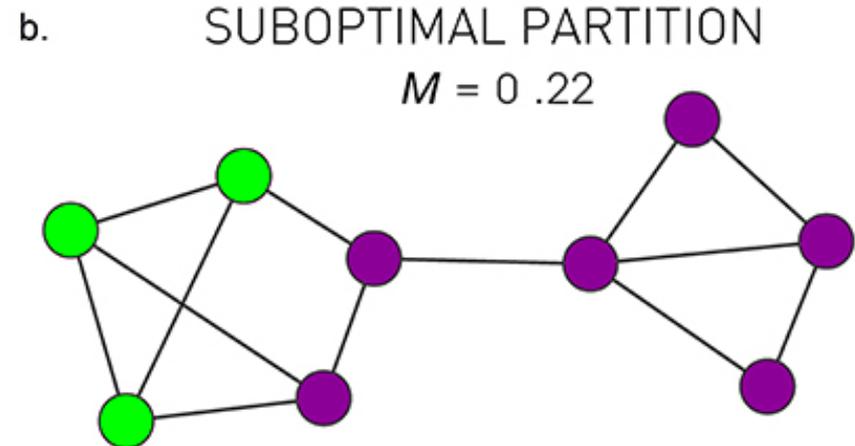
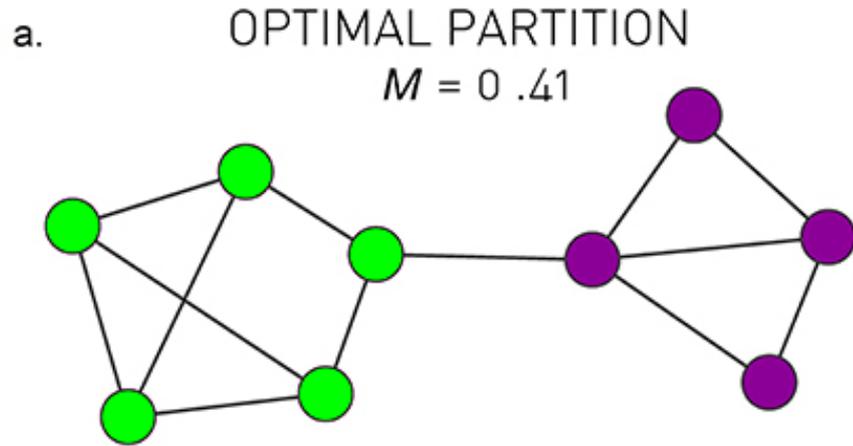
- onde m_k é o número de arestas dentro da parte k , d_k é a soma dos graus dos vértices da parte k

Propriedades

- $M = 0$ se $k=1$: a rede inteira define um *baseline*
- $M < 0$ se $k=n$: muitas partições é ruim
- $\max M \leq 1$: valor máximo da modularidade é 1

Partição é chamado de comunidades em *network science*

Exemplo de Modularidade



Calculando Modularidade

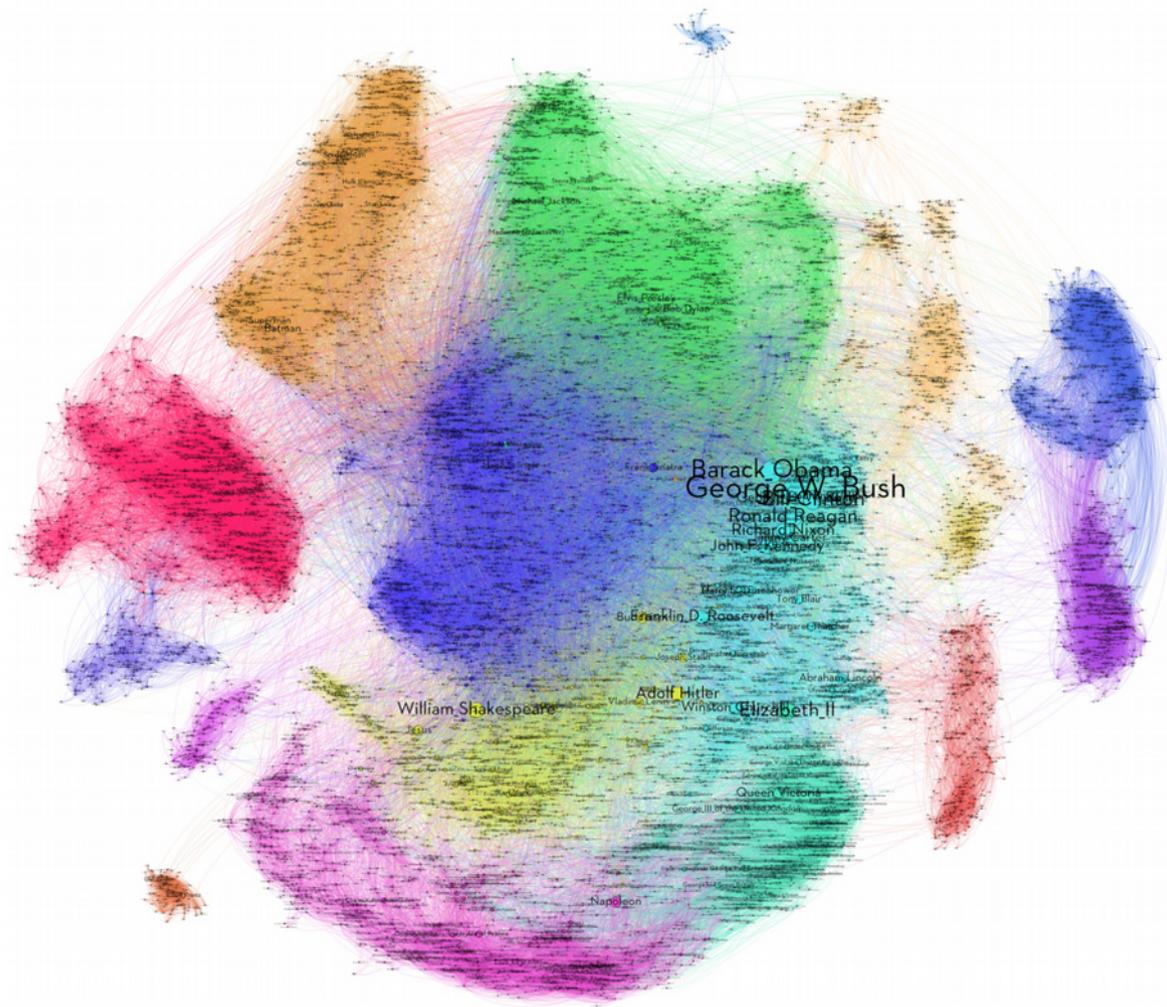
- **Problema:** determinar partição que maximiza modularidade de uma rede
 - não se conhece algoritmo eficiente
- Algoritmo guloso (heurística) de Newman
 - 1) Cada vértice em sua parte
 - 2) Determinar ganho de modularidade ao juntar duas partes (para todos os pares de partes)
 - 3) Fundir partes que possui maior ganho
 - 4) Voltar ao passo 2 até ter uma única parte
 - 5) Retornar partição de maior modularidade (guardar partição e modularidade a cada passo)
- Complexidade (cada passo): $O((n+m)n)$

Algoritmo de Louvain

- Proposto por pesquisadores de UC Louvain, Bélgica (Blondel et al.)
 - algoritmo guloso (heurística) ainda mais rápido
 - 1) Cada vértice em sua parte
 - 2) Determinar ganho de modularidade ao mover vértice para parte de um de seus vizinhos
 - 3) Fazer fusão de maior ganho, se houver
 - 4) Construir nova rede onde cada parte é um vértice
 - 5) Voltar ao passo 2 até não conseguir melhorar
- Complexidade (cada passo): $O(m)$

Algoritmo de Louvain

- Funciona em redes com 100 milhões de vértices (bilhões de arestas)
- Algoritmo pode ser paralelizado



- Comunidades de uma rede social
- Cada cor é uma comunidade (parte) identificada pelo algoritmo

Limitações de Modularidade

- Métrica é enviesada para partes maiores
 - partes pequenas reduzem modularidade, quando rede cresce: *resolution limit*
- Diferença de modularidade ao juntar partes A e B

$$\delta M_{AB} = \frac{m_{AB}}{m} - \frac{d_A d_B}{2m^2}$$

- onde m_{AB} é o número de arestas entre A e B, d_A é a soma dos graus dos vértices de A

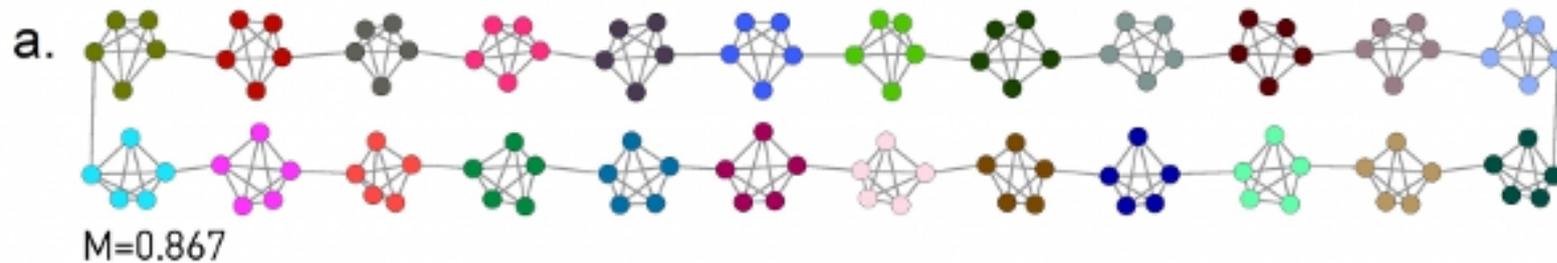
- Se d_A, d_B grande o suficiente, basta $m_{AB} = 1$ para juntar as duas partes!

$$d_A \geq \sqrt{2m}$$

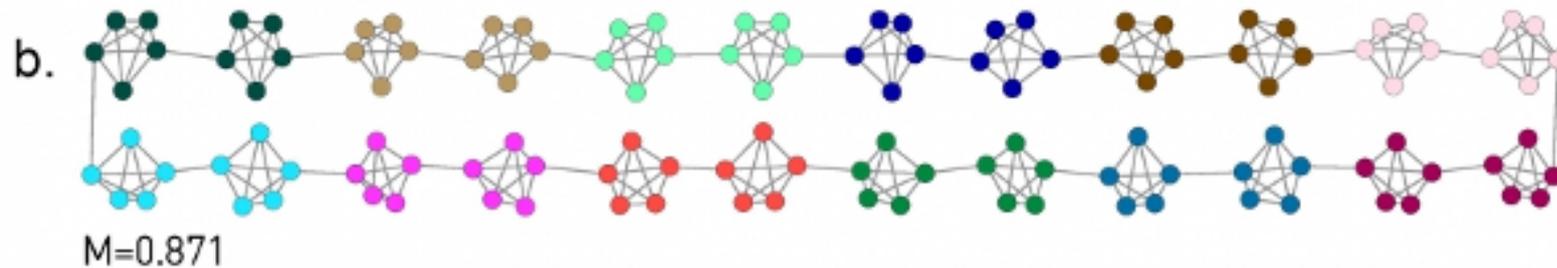
- Condição (aproximada) para tamanho mínimo (em arestas) de cada parte para maximizar modularidade da rede

Limitações de Modularidade

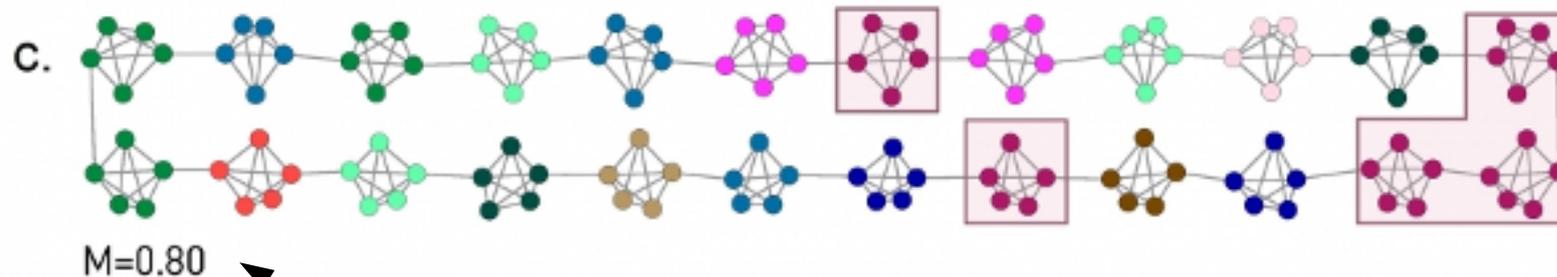
■ Anel com 24 cliques de tamanho 5



■ Partição intuitiva (cada clique é uma parte)



■ Modularidade máxima (duas cliques por parte)



■ Partição aleatória (cliques em partes)

Valor relativamente alto!

Partição com Sobreposição

- Vértices frequentemente pertencem a diferentes comunidades (partes)
 - redes sociais: família, trabalho, escola, hobby
 - redes de informação: diferentes assuntos
- Partição com sobreposição
 - vértice pode estar em mais de uma parte (comunidade)
- Algoritmos
 - CFinder: baseado em percolação de cliques
 - Link Clustering: baseado em similaridade arestas

Ainda é tema de pesquisa!