Hľadanie komunít v sieťach II

Beáta Stehlíková

2-EFM-155 Analýza sociálnych sietí

Fakulta matematiky, fyziky a informatiky, UK v Bratislave, 2020

Hľadanie komunít v sieťach

Prístup I

- Kliky
- Jadrá

Prístup II

- *komunita* bez presnej definície, približne: veľa hrán medzi sebou, málo s inými komunitami
- funkcie napríklad `cluster_walktrap`, `cluster_leading_eigenvector` a pod.

Teraz

- vysvetlíme si, čo je *modularita* a ako súvisí s meraním kvality zhlukovania vrcholov do komunít
- ukážeme si optimalizáciu modularity a pár ďalších algoritmov
- zhrnieme si prácu s komunitami - obrázky, výpočet modularity, možnosť voľby počtu komunít

Modularita

Definícia

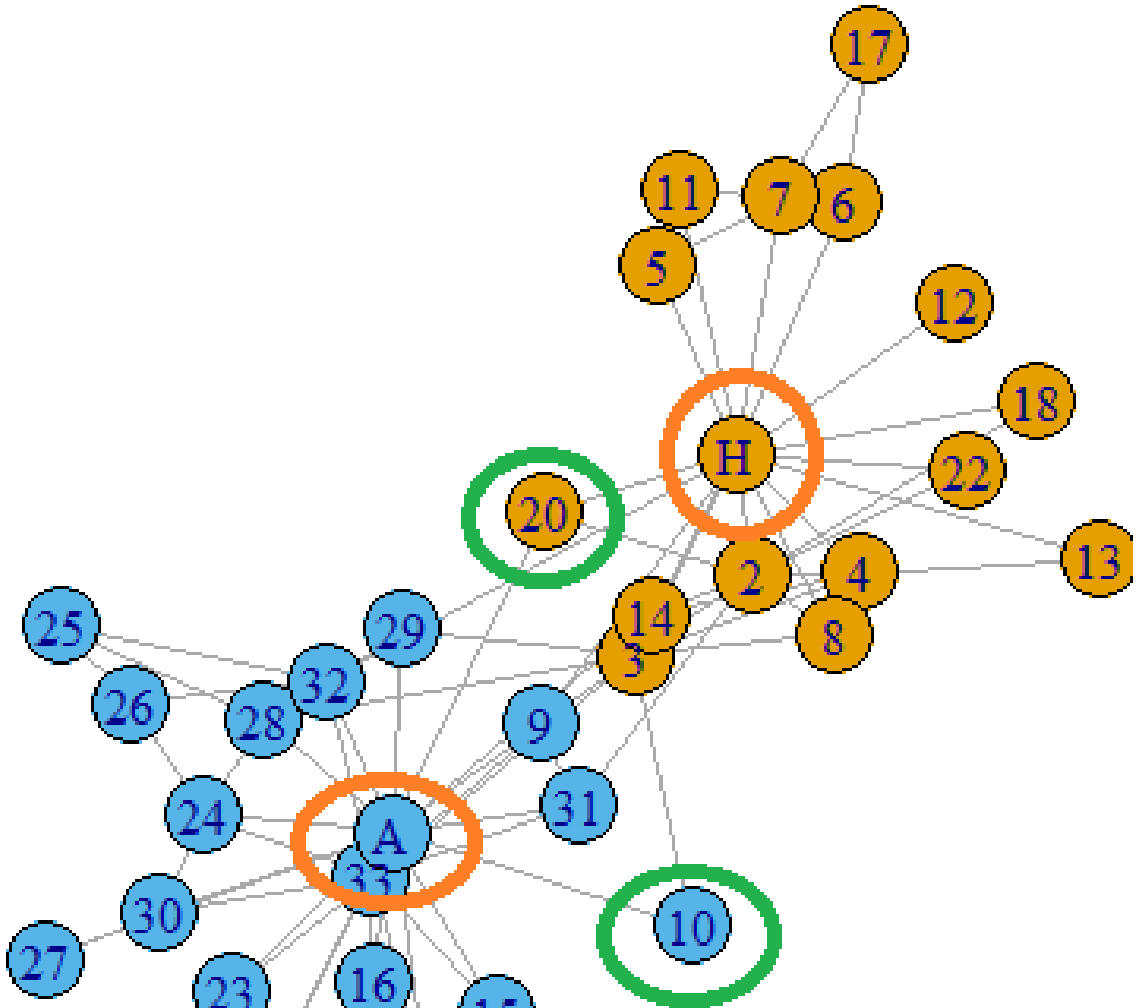
- Pracujeme s neorientovanými sieťami, ak sú vážené - väčšia váha znamená silnejšiu väzbu medzi vrcholmi
- Odvodenie spravíme pre neväžené siete, ozn. $A =$ matica susednosti, $m =$ počet hrán
- Inituitívne: modularita bude *(počet hrán vrámci komunít) mínus (očakávaný počet hrán vrámci komunít)* a budeme ju maximalizovať
- Treba upresniť, čo znamená *očakávaný počet* (teda na základe *akého modelu* ide o očakávaný počet)
- Nech P_{ij} je pravd., že vznikne hrana medzi vrcholmi i a j
- Definujme modularitu ako

$$Q = \frac{1}{2m} \sum_{i,j} [A_{ij} - P_{ij}] \delta(g_i, g_j),$$

kde g_i je komunita, do ktorej patrí vrchol i a $\delta(x, y)$ je 1 pre $x = y$ a inak 0. Optimálne zhlukovanie ju bude maximalizovať.

Model Erdősa a Rényiho

- Voľba tohto modelu by znamenala, že $P_{ij} = p$ (konštanta).
- Lepšie: očakávaný počet hrán sa rovná pozorovanému počtu
- Karate - napr. inštruktor nemá vysoký stupeň “iba náhodou”



Lepšia voľba

- Označme $k_i = degree_i$ stupeň vrchola i
- Očakávaný stupeň každého vrchola sa bude rovnať pozorovanému stupňu:

$$\sum_j P_{ij} = k_i,$$

- P_{ij} má tvar

$$P_{ij} = f(k_i)f(k_j)$$

- Dostaneme

$$P_{ij} = \frac{k_i k_j}{2m}$$

Výpočet v R-ku

- Funkcia `modularity`

Dá sa použiť:

- pre výstup z algoritmov hľadania komunít
- pre náš vektor zo zaradením vrcholov (číslo komunity, kam patrí)

```
data(kite)
com <- cluster_walktrap(kite)
modularity(com)

com.vektor <- c(rep(1, 7), rep(2, 3))
modularity(kite, com.vektor)
```

Cvičenie: Zobrazte komunity pre `com.vektor` - nakreslite sieť a vrcholy odlíšte farebne podľa komunity. Je toto rozumné delenie vrcholov do komunít? Ako sa to prejaví na modularite?

Výpočet v R-ku

Iný postup pre vlastné komunity - vytvoríme objekt typu `communities` (a pracujeme s ním ďalej ako s výstupom z napríklad `cluster_walktrap`):

```
com2 <- make_clusters(kite, membership = com.vektor)
class(com2)
```

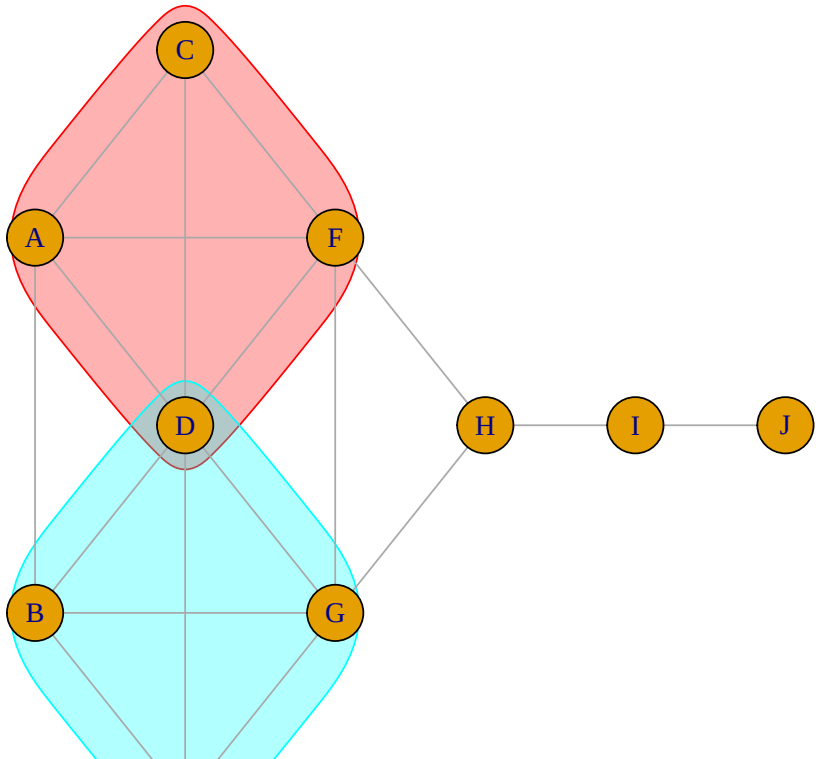
```
## [1] "communities"
```

Potom napr. `plot(com2, kite)`

Vyznačenie skupiny vrcholov v R-ku

Môže byť niekedy užitočné - nevyznačujeme priamo komunity, ale chceme zvýrazniť niektoré vrcholy

```
plot(kite, mark.groups = list(V(kite)[c("A", "C", "D", "F")],
                              V(kite)[c("B", "D", "G", "E")]
```



Optimalizácia modularity

- Metóda `cluster_optimal` priamo optimalizuje modularitu
- V rozumnom čase zbehne len pre menšie siete
- Prečo: rieši sa úloha celočíselného lineárneho programovania, počet premenných je rádu n^2 (n je počet vrcholov siete), premenné sú X_{ij} vyjadrujúce, či sú vrcholy i a j v tom istom zhluku (hodnota 1) alebo nie (hodnota 0)

Optimalizácia modularity

Z pôvodného článku *U. Brandes et al.: On Finding Graph Clusterings with Maximum Modularity*:

The problem of maximizing modularity can be cast into a very simple and intuitive integer linear program (ILP). Given a graph $G = (V, E)$ with $n := |V|$ nodes, we define n^2 decision variables $X_{uv} \in \{0, 1\}$, one for every pair of nodes $u, v \in V$. The key idea is that these variables can be interpreted as an equivalence relation (over V) and thus form a clustering. In order to ensure consistency, we need the following constraints, which guarantee

$$\text{reflexivity } \forall\, u\colon X_{uu} = 1 \ ,$$

$$\text{symmetry } \forall\, u, v\colon X_{uv} = X_{vu} \ , \text{ and}$$

$$\text{transitivity } \forall\, u, v, w\colon \begin{cases} X_{uv} + X_{vw} - 2 \cdot X_{uw} \leq 1 \\ X_{uw} + X_{uv} - 2 \cdot X_{vw} \leq 1 \\ X_{vw} + X_{uw} - 2 \cdot X_{uv} \leq 1 \end{cases} .$$

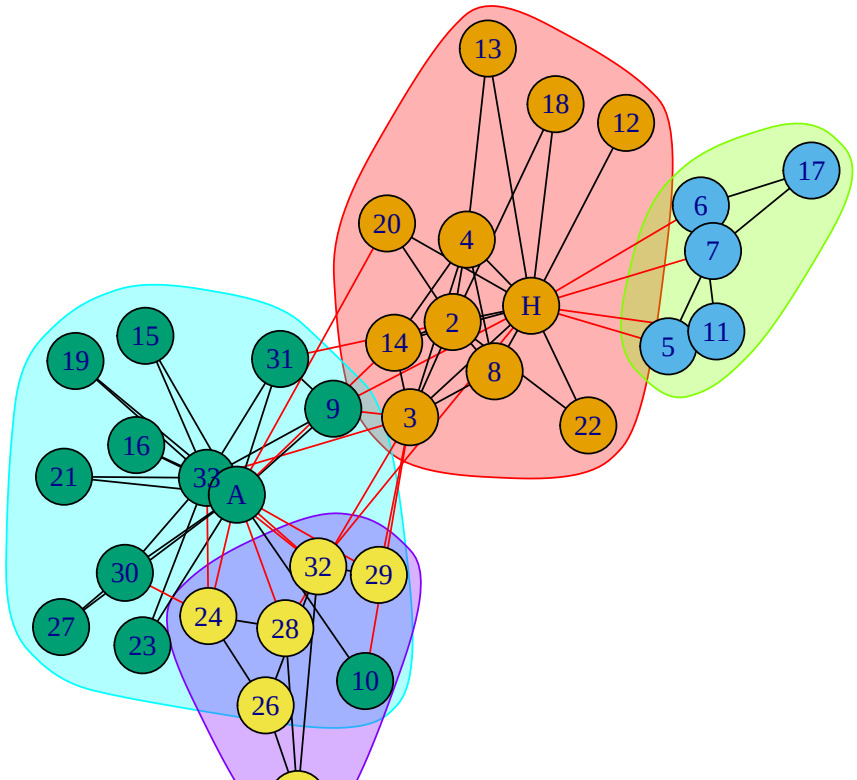
The objective function of modularity then becomes

$$\frac{1}{2m} \sum_{(u,v) \in V^2} \left(E_{uv} - \frac{\deg(u) \deg(v)}{2m} \right) X_{uv} \ ,$$

$$\text{with } E_{uv} = \begin{cases} 1 & \text{, if } (u, v) \in E \\ 0 & \text{, otherwise} \end{cases} .$$

Optimalizácia modularity: príklad

```
library(igraphdata)
data(karate)
komunity <- cluster_optimal(karate)
plot(komunity, karate)
```



Motivácia pre tvorbu iných algoritmov

- Potrebujeme zhlukovanie aj pre väčšie (aj pre veľmi veľké) siete
- Optimalizácia modularity nemá vždy "ideálne" vlastnosti - pozrime sa na príklady z citovaného článku:
  - Príklad 1: pridáme jeden vrchol a výrazne to zmení zhlukovanie (a navyše vrchol, ktorému sa nezmenia susedia, je po novom zaradený do iného zhluku)
  - Príklad 2: neintuitívny výsledok
  - Príklad 3: Ak vytvoríme sieť z dvoch identických komponentov, výsledné zhlukovanie nemusí zodpovedať zhlukovaniu každého z komponentov samostatne
- Môžeme chcieť zadať vlastný počet zhlukov (`cluster_optimal` to neumožňuje)

Motivácia - príklad 1

Non-Locality. At a first view, modularity seems to be a local quality measure. Recalling Equation (1), each cluster contributes separately. However, the example presented in Figures 1(a) and 1(b) exhibit a typical non-local behavior. In these figures, clusters are represented by colors. By adding an additional node connected to the leftmost node, the optimal clustering is altered completely. According to Lemma 2 the additional node has to be clustered together with the leftmost node. This leads to a shift of the leftmost white node from the white cluster to the black cluster, although locally its neighborhood structure has not changed.

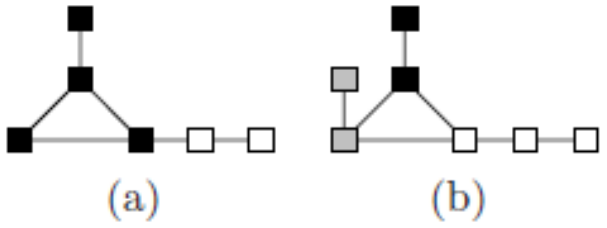
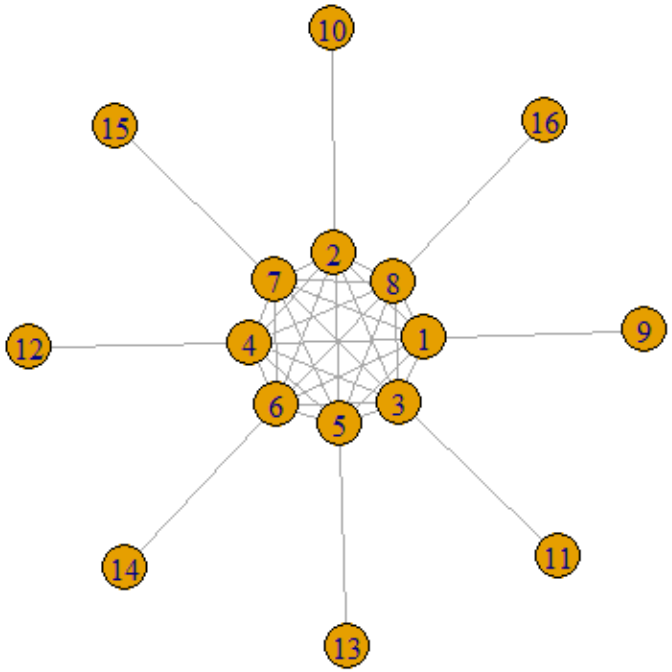
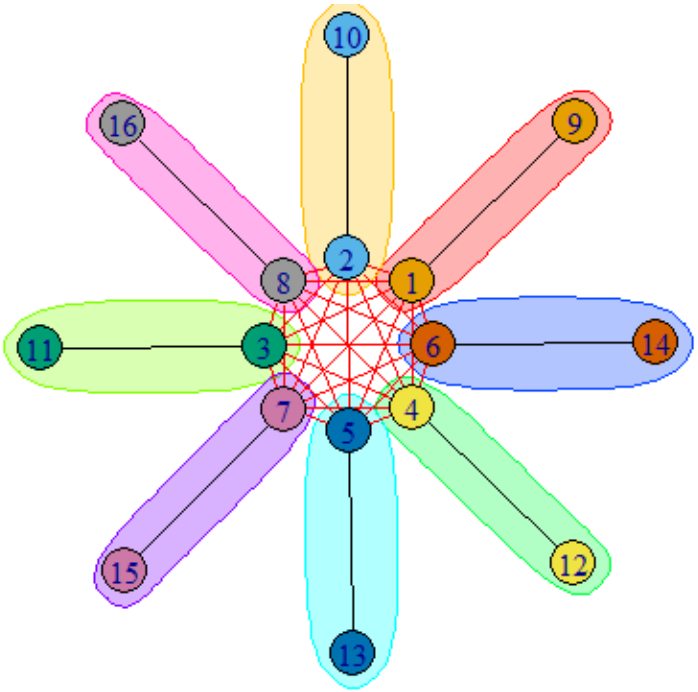


Fig. 1. Non-local behavior. Clusters are represented by colors.

Motivácia - príklad 2

Sensitivity to Satellites. A *clique with leaves* is a graph of $2n$ nodes that consists of a clique K_n and n *leaf* nodes of degree one, such that each node of the clique is connected to exactly one leaf node. For a clique, the trivial clustering with $k = 1$ has maximum modularity. For a clique with leaves, however, the optimal clustering changes to $k = n$ clusters, in which each cluster consists of a connected pair of leaf and clique nodes.





Motivácia - príklad 3

Scaling Behavior. Figures 2(a) and 2(b) display the scaling behavior of modularity. By simply doubling the graph presented in Figure 2(a), the optimal clustering is altered completely. While in Figure 2(a) we obtain three clusters each consisting of the minor K_2, the clustering with maximum modularity of the graph in Figure 2(b) consists of two clusters, each being a graph equal to the one in Figure 2(a).

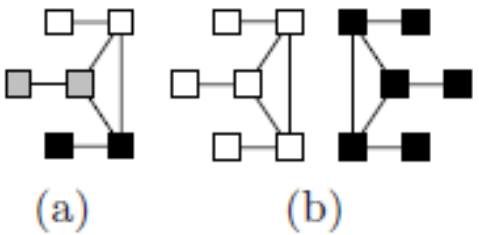


Fig. 2. Scaling behavior. Clustering by colors.

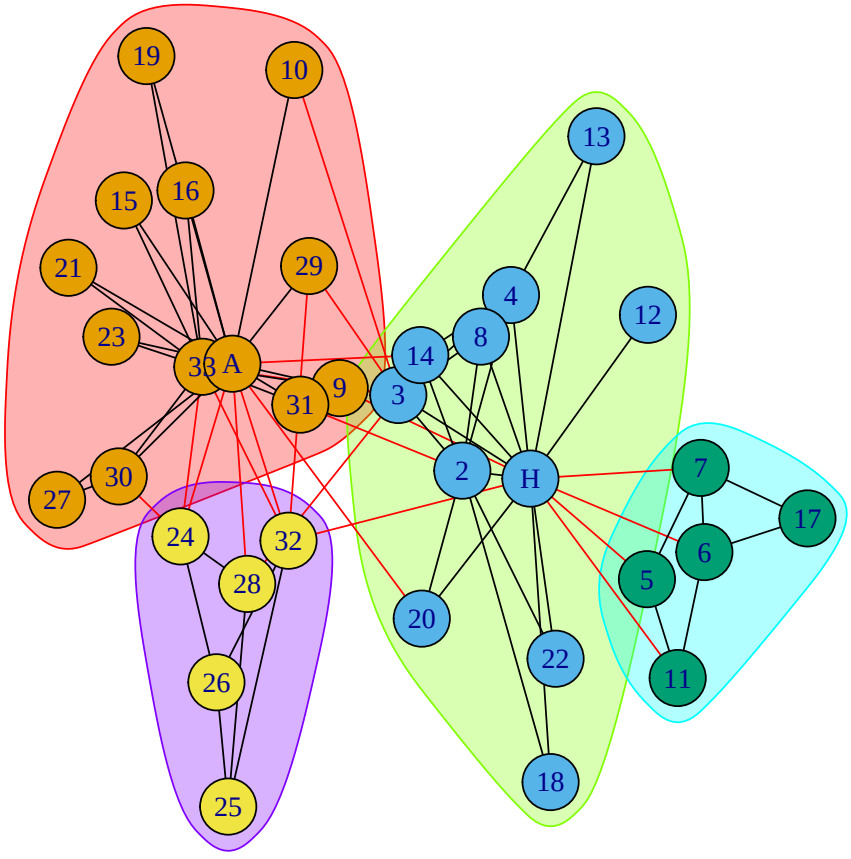
Walktrap algoritmus

Walktrap algoritmus

- *walk* - prechádzka, *trap* - pasca
- pri náhodnej prechádzke po vrcholoch siete by sme mali zostať s veľkou pravdepodobnosť v tej istej komunite
- v R-ku `cluster_walktrap`

Walktrap - príklad

```
komunity  <- cluster_walktrap(karate)
plot(komunity , karate)
```



Walktrap - náš počet zhlukov

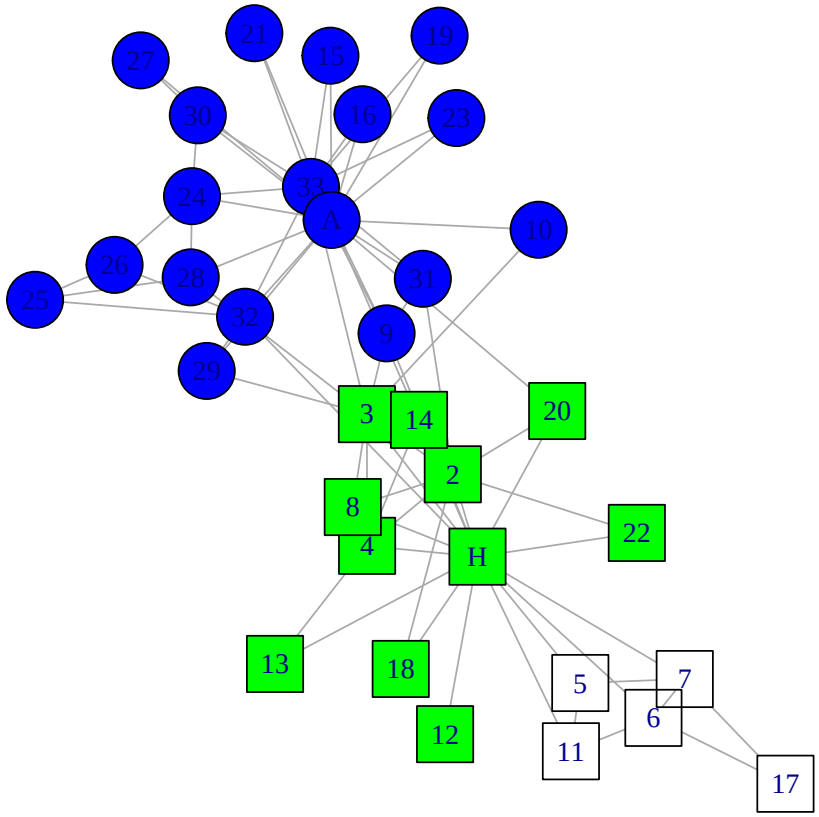
```
nase_rozdelenie_3 <- cut_at(komunity, no = 3) # chceme tri
nase_rozdelenie_3
```

```
##  [1] 2 2 2 2 3 3 3 2 1 1 3 2 2 2 1 1 3 2 1 2 1 2 1 1 1 1
```

Takže napríklad:

```
plot(karate,
     vertex.color = c("blue", "green")[nase_rozdelenie_3],
     vertex.shape = c("square", "circle")[V(karate)$
                                          Faction])
```

Walktrap - náš počet zhlukov



Walktrap - náš počet zhlukov

Porovnajme modularitu:

```
for (i in 2:6){
  nase_rozdelenie <- cut_at(komunity, i)
  print(modularity(karate, nase_rozdelenie))
}
```

```
## [1] 0.3714661
## [1] 0.3990796
## [1] 0.4111604
## [1] 0.398833
## [1] 0.3908613
```

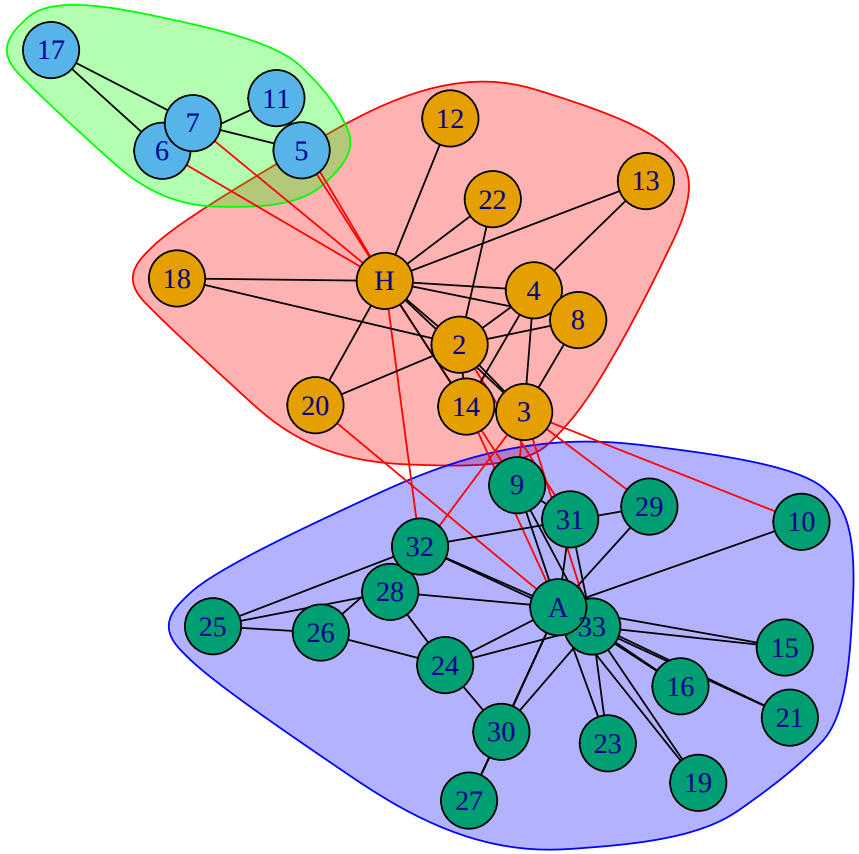
Label propagation algoritmus

Label propagation algoritmus

- `label` - nálepka, značka, `propagation` - šírenie
- vrcholy dostanú značky a potom sa značky menia tak, aby mal vrchol takú značku ako väčšina jeho susedov
- v R-ku: `cluster_label_prop`

Label propagation - príklad

```
komunity  <- cluster_label_prop(karate)
plot(komunity , karate)
```



Label propagation - vlastný počet zhlukov

Počet zhlukov sa v tomto prípade nedá zadať:

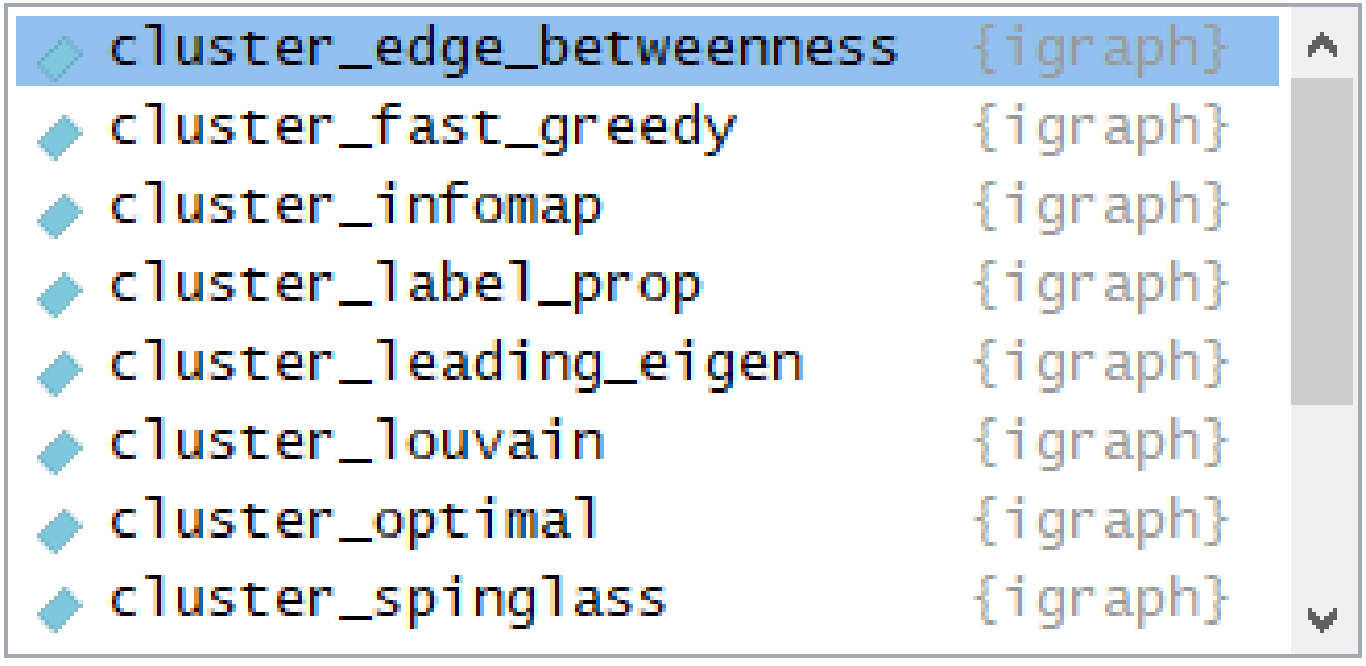
```
> nase_rozdelenie_3 <- cut_at(komunity, no = 3)
Error in cut_at(komunity, no = 3) :
  Not a hierarchical communitity structure
```

Ďalšie algoritmy v R-ku

Ďalšie algoritmy

- Sú aj ďalšie algoritmy:

```
> cluster_
```



- Niektoré umožňujú vlastný počet zhlukov, niektoré nie
- V literatúre stále vznikajú nové algoritmy

Námet na projekt: Zhlukovanie na základe vlastného vektora

Algoritmus M. E. J. Newmana

M. E. J. Newman: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E 74, 036104 (2006)

https://arxiv.org/abs/physics/0605087

Podľa kapitoly 4.A vysvetlite:

- ako súvisia vlastné hodnoty a vektory (a akej matice) s modularitou
- ako z vlastného vektora (a akého) získať rozdelenie vrcholov do dvoch komunít

V R-ku

- funkcia `cluster_leading_eigen` pokračuje ďalších delením (neobmedzuje sa na dve komunity)
- naprogramujte delenie do dvoch komunít podľa algoritmu z článku
- spravte ilustračný príklad z nasledujúcich slajdov, potom zaujímavú aplikáciu, porovnanie s inými algoritmami a pod.

Príklad

Absolútna hodnota vlastného vektora hovorí o tom, "ako silno" patrí vrchol do svojej komunity.

```
set.seed(111)
g <- sample_pa(n=20, directed = FALSE, power=0.5)
g <- add_edges(g, c(1,5, 17,5, 6,4, 6,11,
                    6,8, 12,15, 9, 4))
lay <- layout.fruchterman.reingold(g)
plot(g, layout=lay, vertex.size=10)
```

- Rozdeľte vrcholy do dvoch komunít podľa vlastného vektora
- Vyznačte v grafe farebne komunity (farbou vrcholov alebo obrysmi pomocou `mark.groups`)
- Veľkosť vrcholov nech je úmerná absolútnej hodnote prvku vlastného vektora (teda veľké vrcholy budú "silne zviazané" so svojou komunitou)

Príklad