Hierarchical network clustering by modularity maximization

Sonia Cafieri

Laboratoire MAIAA

ENAC - École Nationale de l'Aviation Civile

University of Toulouse

France

Workshop on Clustering and Search techniques in large scale networks
Nizhny Novgorod, November 2014

Outline

1. Hierarchical network clustering
   - Agglomerative and Divisive heuristics
2. Modularity-based hierarchical clustering
   - Agglomerative modularity heuristics
   - Divisive modularity heuristics

Outline

1. Hierarchical network clustering
   - Agglomerative and Divisive heuristics
2. Modularity-based hierarchical clustering
   - Agglomerative modularity heuristics
   - Divisive modularity heuristics

Hierarchical complex systems

Hierarchy is observed or postulated in many complex systems

- several levels of grouping of the entities $\Rightarrow$ multilevel structure
- different levels of organization/structure at different scales
- partitions can be hierarchically ordered

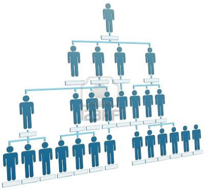
Example

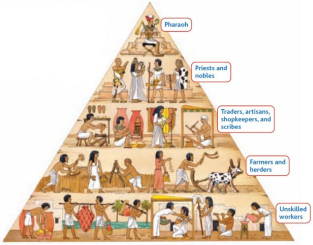
Social network of children living in the same town:

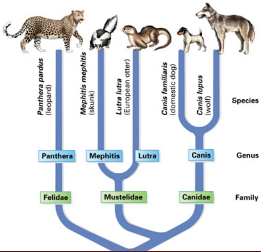
one could group the children according to the schools they attend,

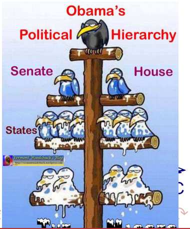
within each school one can make a subdivision into classes, etc.

Hierarchies









Hierarchical graph clustering heuristics

Hierarchical heuristics are in principle devised for

finding a hierarchy of partitions implicit in the given network

They aim at finding a set of nested partitions.

- Agglomerative heuristics
- Divisive heuristics

Outline

1. Hierarchical network clustering
   - Agglomerative and Divisive heuristics

2. Modularity-based hierarchical clustering
   - Agglomerative modularity heuristics
   - Divisive modularity heuristics

Agglomerative and Divisive heuristics

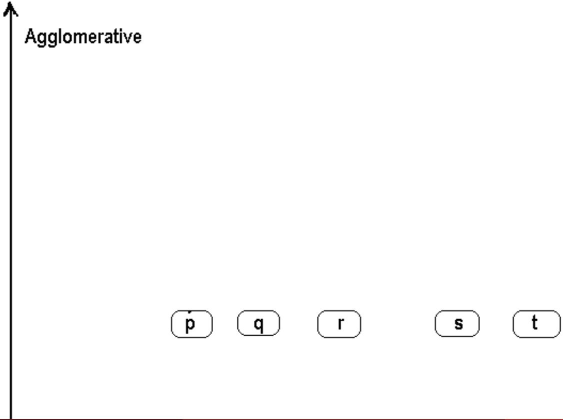
Agglomerative heuristics

- Proceed from an initial partition with n communities each containing 1 entity
- Iteratively merge the pair of entities for which merging increases most the objective function (e.g., modularity)

Agglomerative and Divisive heuristics

Agglomerative heuristics

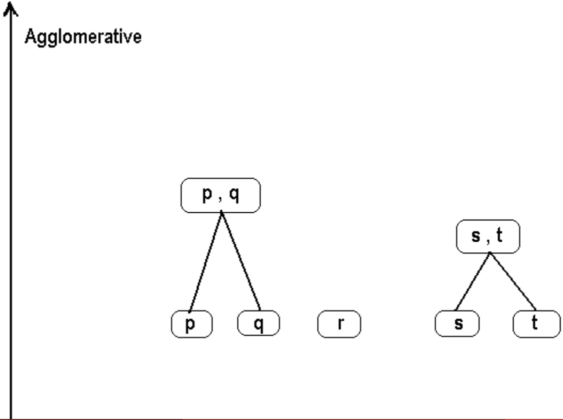
- Proceed from an initial partition with n communities each containing 1 entity
- Iteratively merge the pair of entities for which merging increases most the objective function (e.g., modularity)



Agglomerative and Divisive heuristics

Agglomerative heuristics

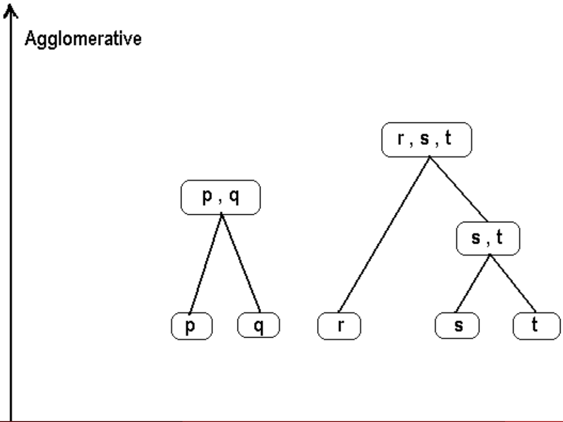
- Proceed from an initial partition with n communities each containing 1 entity
- Iteratively merge the pair of entities for which merging increases most the objective function (e.g., modularity)



Agglomerative and Divisive heuristics

Agglomerative heuristics

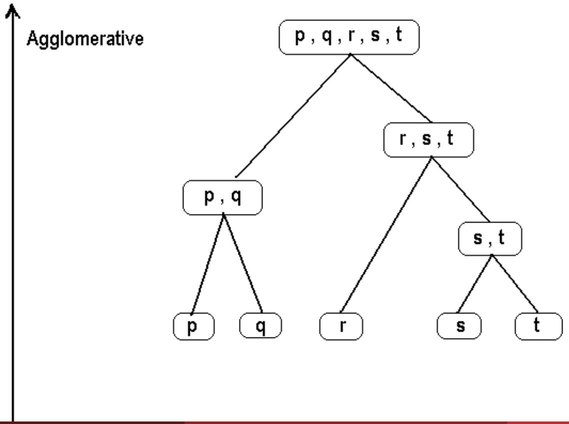
- Proceed from an initial partition with n communities each containing 1 entity
- Iteratively merge the pair of entities for which merging increases most the objective function (e.g., modularity)



Agglomerative and Divisive heuristics

Agglomerative heuristics

- Proceed from an initial partition with n communities each containing 1 entity
- Iteratively merge the pair of entities for which merging increases most the objective function (e.g., modularity)

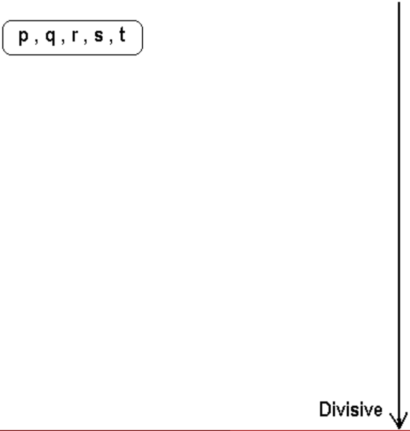


Divisive heuristics

- Proceed from an initial partition containing all entities
- Iteratively divide a cluster into two in such a way to increase most the objective function (or the decrease in the objective value is the smallest possible)

Divisive heuristics

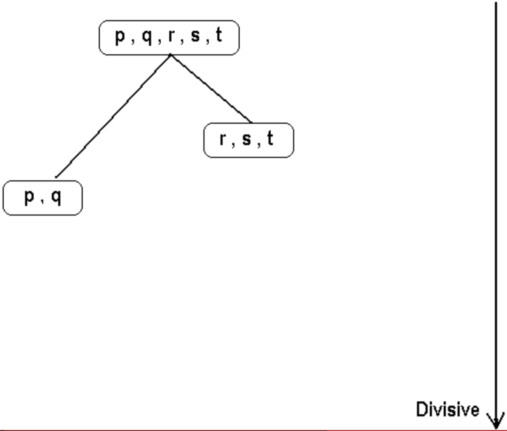
- Proceed from an initial partition containing all entities
- Iteratively divide a cluster into two in such a way to increase most the objective function (or the decrease in the objective value is the smallest possible)



Agglomerative and Divisive heuristics

Divisive heuristics

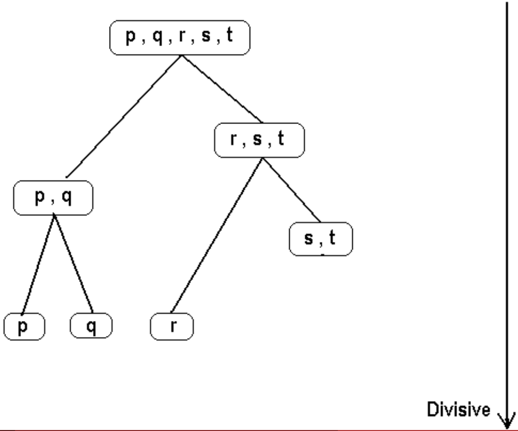
- Proceed from an initial partition containing all entities
- Iteratively divide a cluster into two in such a way to increase most the objective function (or the decrease in the objective value is the smallest possible)



Agglomerative and Divisive heuristics

Divisive heuristics

- Proceed from an initial partition containing all entities
- Iteratively divide a cluster into two in such a way to increase most the objective function (or the decrease in the objective value is the smallest possible)

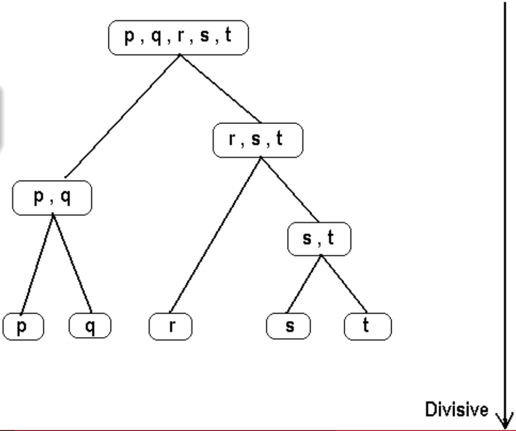


Agglomerative and Divisive heuristics

Divisive heuristics

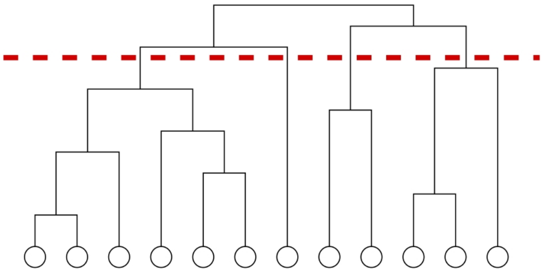
- Proceed from an initial partition containing all entities
- Iteratively divide a cluster into two in such a way to increase most the objective function (or the decrease in the objective value is the smallest possible)

critical point: bipartitioning a cluster



Hierarchical heuristics

Bottom-up and Top-down procedures illustrated by means of dendrograms:



horizontal cuts correspond to partitions of the graph in communities

Sometimes, stopping conditions are imposed to select a partition or a group of partitions satisfying a special criterion:
- a given number of clusters
- the optimization of a quality function (e.g. modularity).

Hierarchical heuristics

- Advantages:
  - does not require a preliminary knowledge on the number and size of the clusters
  - specially suitable for hierarchical systems

- Disadvantages:
  - does not provide a way to discriminate between the obtained partitions
  - the results depend on the specific similarity measure adopted
  - yields a hierarchical structure by construction, which is rather artificial for graphs not having a hierarchical structure

Hierarchical agglomerative and divisive

Agglomerative

- choosing at each iteration which pair of communities should be merged is easy: consider all $O(n^2)$ mergings of pairs of entities
- a careful use of data structures often reduces complexity

Divisive

- finding a bipartition locally optimizing the adopted criterion is more difficult (example: modularity is NP-hard even for 2 clusters)
- bipartitioning requires a specific algorithm

In both cases, no guarantee that the partitions are optimal

Outline

Modularity

Newman and Girvan, 2004:

compare the fraction of edges falling within communities
to the expected fraction of such edges

Modularity:

$$Q = \sum_s [a_s - e_s]$$

a_s = fraction of all edges in module s

e_s = expected value of the same quantity in a graph with same vertex degree and edges placed at random

- $Q \approx 0$: the network is equivalent to a random network (barring fluctuations)
- $Q \approx 1$: the network has a strong community structure
- in practice, max Q often between 0.3 and 0.7

Maximizing modularity gives an optimal partition with the optimal number of clusters

Modularity: another expression

Modularity as a sum of values over all edges of the complete graph K_n:

$$Q = \frac{1}{2m} \sum_{i,j \in V} \left(a_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

where:

- $m = |E|$
- k_i, k_j = degrees of vertices i and j
- a_{ij} = ij component of the adjacency matrix of G
- $\delta(c_i, c_j)$ = 1 if the communities to which i and j belong are the same, 0 otherwise (Kronecker symbol)
- $k_i k_j / 2m$ = expected number of edges between vertices i and j in a null model where edges are placed at random and the distribution of degrees remains the same.

Outline

1. Hierarchical network clustering
   - Agglomerative and Divisive heuristics

2. Modularity-based hierarchical clustering
   - Agglomerative modularity heuristics
   - Divisive modularity heuristics

Building agglomerative modularity heuristics

- Usually greedy
- Decision which clusters should be merged based on:

  cluster C, cluster C' which results from the merge of C_i and C_j of C

  $\Delta Q(C_i, C_j) = Q(C, G) - Q(C', G) = e_{ij} + e_{ji} - 2a_i a_j = 2(e_{ij} - a_i a_j)$

  local measure as it depends only on C_i and C_j:

  e_{ij}= fraction of edges connecting C_i and C_j

  a_i= fraction of edges attached to vertices in C_i

Building agglomerative modularity heuristics

- Usually greedy
- Decision which clusters should be merged based on:

  cluster C, cluster C' which results from the merge of C_i and C_j of C

  $\Delta Q(C_i, C_j) = Q(C, G) - Q(C', G) = e_{ij} + e_{ji} - 2a_i a_j = 2(e_{ij} - a_i a_j)$

  local measure as it depends only on C_i and C_j:

  e_{ij}= fraction of edges connecting C_i and C_j

  a_i= fraction of edges attached to vertices in C_i

Question

How to select clusters to be merged?

Existing agglomerative modularity heuristics

- Newman, 2004:

  At each step, two clusters C_i and C_j get merged that have the highest $\Delta Q(C_i, C_j)$.

  Slow, as $\Delta Q(C_i, C_j)$ computed for each pair of communities.

Existing agglomerative modularity heuristics

- Newman, 2004:

  At each step, two clusters C_i and C_j get merged that have the highest $\Delta Q(C_i, C_j)$.

  Slow, as $\Delta Q(C_i, C_j)$ computed for each pair of communities.

- Clauset-Newman-Moore, 2004 (CNM):

  $\Delta Q(C_i, C_j)$ only recalculated if there is at least an edge joining C_i and C_j.

  Careful use of data structures is done.

  Significantly faster than Newman's heuristic.

Existing agglomerative modularity heuristics

- Newman, 2004:

  At each step, two clusters C_i and C_j get merged that have the highest $\Delta Q(C_i, C_j)$.

  Slow, as $\Delta Q(C_i, C_j)$ computed for each pair of communities.

- Clauset-Newman-Moore, 2004 (CNM):

  $\Delta Q(C_i, C_j)$ only recalculated if there is at least an edge joining C_i and C_j.

  Careful use of data structures is done.

  Significantly faster than Newman's heuristic.

- Schuetz and Caflisch, 2008 (MSG):

  multistep greedy algorithm, builds classes of joins (= pairs of vertices) with the same $\Delta Q(C_i, C_j)$ and sorts them in descending order. In each step all joins in the top l classes are executed.

  Faster than CNM.

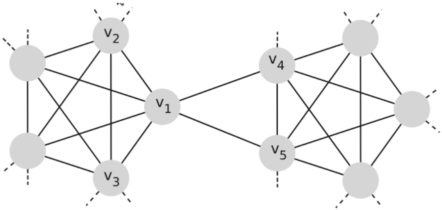
Agglomerative modularity heuristics: Remarks

- Prior mergers in the neighborhood of a cluster influence later merger decisions for this cluster
- Possibly unbalanced merge processes, where some regions of the graph are heavily more contracted than others
  $\Rightarrow$ bad clustering results

Agglomerative modularity heuristics: Remarks

- Prior mergers in the neighborhood of a cluster influence later merger decisions for this cluster
- Possibly unbalanced merge processes, where some regions of the graph are heavily more contracted than others
  $\Rightarrow$ bad clustering results

Example



1) merging $C_i = \{v_1\}$ and $C_j = \{v_4\}$:

$$e_{ij} = 1, \quad a_i = 6, \; a_j = 6$$

$$\Delta Q = 2\left(\frac{1}{2m} - \frac{6}{2m}\frac{6}{2m}\right) = \frac{2}{2m}\left(1 - \frac{6*6}{2m}\right)$$

2) merging $C_i = \{v_1\}$ and $C_j = \{v_4, v_5\}$:

$$e_{ij} = 2, \quad a_i = 6, \; a_j = 12$$

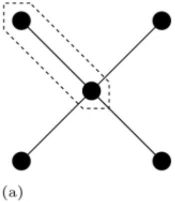
$$\Delta Q = 2\left(\frac{2}{2m} - \frac{6}{2m}\frac{12}{2m}\right) = \frac{4}{2m}\left(1 - \frac{6*6}{2m}\right)$$

Agglomerative modularity heuristics: Remarks

Example 2 Star-like graph

Agglomerative modularity heuristics: Remarks

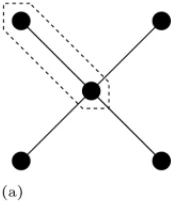
Example 2 Star-like graph



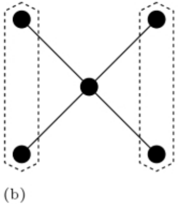
(a) merging two vertices

Agglomerative modularity heuristics: Remarks

Example 2 Star-like graph



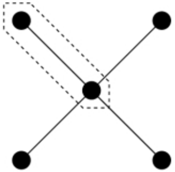
(a) merging two vertices



(b) merging vertices with the same neighbours

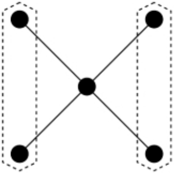
Agglomerative modularity heuristics: Remarks

Example 2 Star-like graph



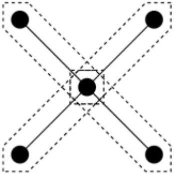
(a)

(a) merging two vertices



(b)

(b) merging vertices with the same neighbours



(c)

(c) merging more than two vertices at a time

ENAC

Outline

Building divisive modularity heuristics

Question

What we need to build a divisive algorithm?

Building divisive modularity heuristics

Question

What we need to build a divisive algorithm?

Two subproblems:

- Select the cluster to split (bipartition)
- Solve the bipartitioning problem

Building divisive modularity heuristics

Question

What we need to build a divisive algorithm?

Two subproblems:

- Select the cluster to split (bipartition)
- Solve the bipartitioning problem

Question

When using modularity?

Existing divisive modularity heuristics (1/2)

Finding the optimal (modularity maximizing) splitting:

- Newman, 2006 (*spectral*):

  The first eigenvector of the modularity matrix $B = (b_{ij})$ with

  $$b_{ij} = a_{ij} - k_i k_j / 2m$$

  is computed. The entities corresponding to positive components of this eigenvector form one community and the remaining ones form the other.

Existing divisive modularity heuristics (1/2)

Finding the optimal (modularity maximizing) splitting:

- Newman, 2006 (*spectral*):

  The first eigenvector of the modularity matrix $B = (b_{ij})$ with

  $$b_{ij} = a_{ij} - k_i k_j / 2m$$

  is computed. The entities corresponding to positive components of this eigenvector form one community and the remaining ones form the other.

- Kernighan-Lin heuristic (*KL*):

  from an initial bipartition, proceed to a sequence of reassignments of one entity from a community to the other.

  - At each step, select and perform the reassignment which improves most, or deteriorates least, the objective function value (modularity) ;

    further reassignments of the moved entity are forbidden.
  - Once no more reassignments are allowed, select the best partition found among the considered partitionsas new initial partition.
  - Stops the whole procedure when a full sequence of n reassignments does not lead to any improvement.

Existing divisive modularity heuristics (2/2)

- Newman, 2006: *spectral* + *KL*:

  KL used as refinement step

Existing divisive modularity heuristics (2/2)

- Newman, 2006: *spectral* + *KL*:
  KL used as refinement step
- Cafieri et al., 2011 (*CHL*):
  Bipartition is computed exactly solving a mixed-integer quadratic problem (MIQP), with a convex continuous relaxation.
  Modularity as objective function of the MIQP

MIQP for modularity maximization (Xu, Tsoka and Papageorgiou, 2007)

Variables used to identify to which module each vertex and each edge belongs:

$$X_{rs} = \begin{cases} 1 & \text{if edge } r \text{ belongs to module } s \\ 0 & \text{otherwise} \end{cases} \qquad \forall r = 1, 2, \dots m, \; s = 1, 2, \dots M$$

$$Y_{is} = \begin{cases} 1 & \text{if vertex } i \text{ belongs to module } s \\ 0 & \text{otherwise.} \end{cases} \qquad \forall i = 1, 2, \dots n, \; s = 1, 2, \dots M$$

$$\max Q = \sum_s [a_s - e_s] = \sum_s \left[\frac{m_s}{m} - \left(\frac{d_s}{2m} \right)^2 \right]$$

m_s = number of edges in module s
d_S = sum of degrees k_i of vertices in s

- $m_s = \sum_r X_{rs}$ and $d_S = \sum_i k_i Y_{is}$
- $\sum_s Y_{is} = 1 \quad \forall i = 1, 2, \dots n$ — each vertex belongs to one module
- $X_{rs} \leq Y_{is} \quad \forall r = \{v_i, v_j\} \in E$
  $X_{rs} \leq Y_{js} \quad \forall r = \{v_i, v_j\} \in E$ — any edge $r = \{v_i, v_j\}$ can belong to module s $\Leftrightarrow$ both of its end vertices i,j belong to s
- $u_s \leq u_{s-1}$ — module s nonempty $\Leftrightarrow$ $s-1$ is so ($u_s = 1$ if module s nonempty, 0 otherwise)
- symmetry-breaking constraints

MIQP for modularity maximization (Xu, Tsoka and Papageorgiou, 2007)

Variables used to identify to which module each vertex and each edge belongs:

$$X_{rs} = \begin{cases} 1 & \text{if edge } r \text{ belongs to module } s \\ 0 & \text{otherwise} \end{cases} \qquad \forall r = 1, 2, \ldots m,\ s = 1, 2, \ldots M$$

$$Y_{is} = \begin{cases} 1 & \text{if vertex } i \text{ belongs to module } s \\ 0 & \text{otherwise.} \end{cases} \qquad \forall i = 1, 2, \ldots n,\ s = 1, 2, \ldots M$$

$$\max Q = \sum_s [a_s - e_s] = \sum_s \left[\frac{m_s}{m} - \left(\frac{d_s}{2m} \right)^2 \right]$$

m_s = number of edges in module s
d_S = sum of degrees k_i of vertices in s

- $m_s = \sum_r X_{rs}$ and $d_S = \sum_i k_i Y_{is}$
- $\sum_s Y_{is} = 1 \quad \forall i = 1, 2, \ldots n$
- $X_{rs} \leq Y_{is} \quad \forall r = \{v_i, v_j\} \in E$
  $X_{rs} \leq Y_{js} \quad \forall r = \{v_i, v_j\} \in E$
- $u_s \leq u_{s-1}$
- symmetry-breaking constraints

$\Downarrow$

Mixed-Integer Quadratic Program

with a convex continuous relaxation

An exact algorithm for bipartition

$$Q = \sum_s \left[\frac{m_s}{m} - \left(\frac{d_s}{2m} \right)^2 \right] \qquad \text{bipartition} \;\Rightarrow\; s \in \{1, 2\}$$

Express d_2 as a function of d_1: $d_2 = d_t - d_1$
(d_t = sum of degrees in the community to be bipartitioned)

$\Rightarrow$ Modularity: $Q = \dfrac{m_1 + m_2}{m} - \dfrac{d_1^2}{4m^2} - \dfrac{d_t^2}{4m^2} + \dfrac{d_t d_1}{2m^2}$

An exact algorithm for bipartition

$$Q = \sum_s \left[\frac{m_s}{m} - \left(\frac{d_s}{2m} \right)^2 \right]$$

bipartition $\Rightarrow s \in \{1, 2\}$

the MIQP can be specialized

Express d_2 as a function of d_1: $d_2 = d_t - d_1$

(d_t = sum of degrees in the community to be bipartitioned)

$\Rightarrow$ Modularity: $$Q = \frac{m_1 + m_2}{m} - \frac{d_1^2}{4m^2} - \frac{d_t^2}{4m^2} + \frac{d_t d_1}{2m^2}$$

Bipartitioning model:

$$\begin{cases} \max Q & & & \\ X_{r1} & \leq & Y_{i1} & \forall r = \{v_i, v_j\} \in E \\ X_{r1} & \leq & Y_{j1} & \forall r = \{v_i, v_j\} \in E \\ X_{r2} & \leq & 1 - Y_{i1} & \forall r = \{v_i, v_j\} \in E \\ X_{r2} & \leq & 1 - Y_{j1} & \forall r = \{v_i, v_j\} \in E \\ m_s & = & \sum_r X_{rs} & \forall s \in \{1, 2\} \\ d_1 & = & \sum_{i \in V_1} k_i Y_{i1} & \end{cases}$$

MIQP

CHL hierarchical divisive algorithm

Bipartitioning problem:

Mixed-Integer Quadratic Program

with a single non linear but concave term, in the obj.funct. to be maximized

$\Rightarrow$ continuous relaxation easy to solve $\Rightarrow$ exactly solved using `CPLEX`

Hierarchical divisive algorithm:

- divisive scheme
- splitting step performed using the above exact algorithm for bipartition

$\Longrightarrow$ the proposed heuristic is *locally optimal* (but not globally optimal)

Finding:

the algorithm performs better than the main existing hierarchical algorithms

(*agglomerative* by Clauset *et al.*, *divisive spectral* by Newman)

Closing question

Question

Can cohesion conditions, mixed to modularity, be used to build hierarchical heuristics?

Future research direction!