Mathematical Programming reformulations in modularity maximizing graph clustering

Sonia Cafieri

ENAC – Ecole Nationale de l'Aviation Civile
Toulouse, France

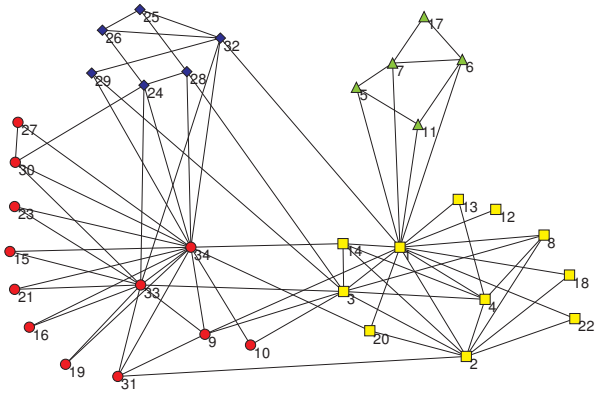
thanks to:

Alberto Costa (Singapore University of Technology and Design)

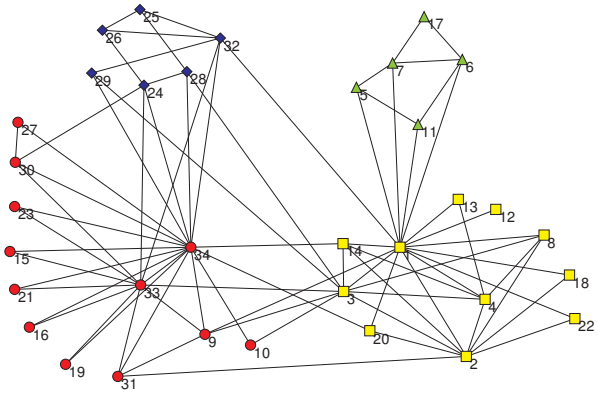
Pierre Hansen (GERAD, HEC, Montréal, Canada)

Summer School on Operational Research and Applications
May 2013

Graph clustering as an optimization problem



Graph clustering as an optimization problem



$$\left.\begin{array}{r} \min f(x) \\ g(x) \leq b \\ x \in X \end{array}\right\} \Rightarrow$$

Mathematical Programming formulation:

defining an optimization problem

Reformulation:

finding a *more convenient* formulation

Aim: starting from a (nonlinear) optimization formulation for a bipartition problem, find new formulations which enhance the efficiency of the proposed solution approach

Outline

Outline

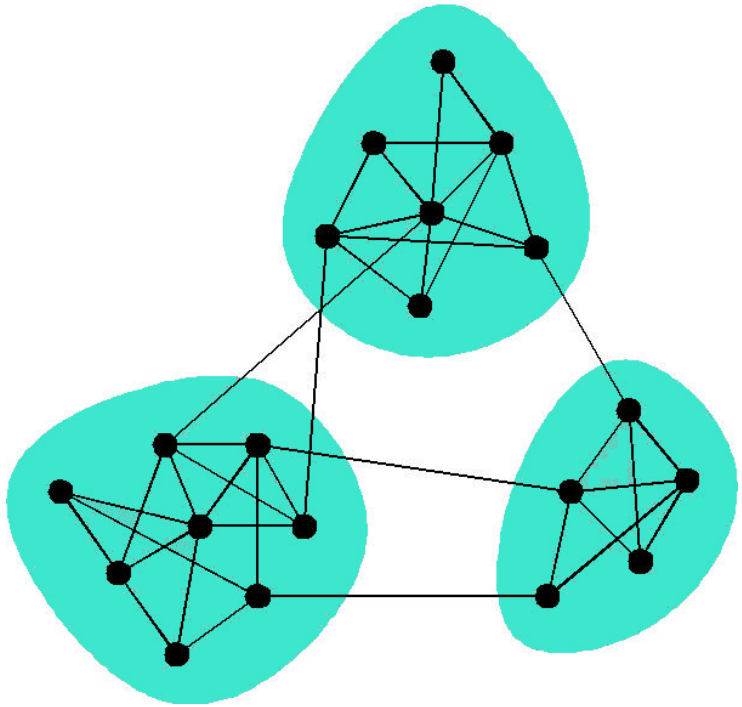
Graph clustering

Automatic analysis of complex systems represented as networks (graphs)

$\Downarrow$

identification of clusters

cluster or **community** = a subset of vertices such that there are more edges within the community than edges joining it to the outside



Outline

Modularity

Modularity [Newman, Girvan; Phys. Rev. E, 2004]

Compare the number of inner edges minus the expected number of such edges in a random graph having the same distribution of degrees of G:

$$Q = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{{D_c}^2}{4m^2} \right)$$

Modularity

Modularity [Newman, Girvan; Phys. Rev. E, 2004]

Compare the number of inner edges minus the expected number of such edges in a random graph having the same distribution of degrees of G:

$$Q = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{{D_c}^2}{4m^2} \right)$$

- N_c: number of clusters;
- m: number of edges of the graph;
- m_c: number of edges in cluster c;
- D_c: sum of degrees of vertices in cluster c;
- $\frac{m_c}{m}$: fraction of edges in cluster c;
- $\frac{{D_c}^2}{4m^2}$: expected number of edges in cluster c in a graph where vertices have same degrees but edges are placed randomly.

Modularity

Ranges of modularity

- $-\frac{1}{2} \leq Q \leq 1$ ($-\frac{1}{2}$ for bipartite graphs, 1 cluster for partition);
- $Q \approx 0$: graph similar to random graph;
- $Q \approx 1$: graph with strong community structure.

Modularity

Ranges of modularity

- $-\frac{1}{2} \leq Q \leq 1$ ($-\frac{1}{2}$ for bipartite graphs, 1 cluster for partition);
- $Q \approx 0$: graph similar to random graph;
- $Q \approx 1$: graph with strong community structure.

How many clusters in the graph?

Modularity

Ranges of modularity

- $-\frac{1}{2} \leq Q \leq 1$ ($-\frac{1}{2}$ for bipartite graphs, 1 cluster for partition);
- $Q \approx 0$: graph similar to random graph;
- $Q \approx 1$: graph with strong community structure.

How many clusters in the graph?

- 1 cluster containing all vertices $\rightarrow Q = 0$;
- n clusters, with 1 vertex for cluster $\rightarrow Q < 0$;
- in general $1 < N_c < n$, not known a priori.

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

Modularity

Ranges of modularity

- $-\frac{1}{2} \leq Q \leq 1$ ($-\frac{1}{2}$ for bipartite graphs, 1 cluster for partition);
- $Q \approx 0$: graph similar to random graph;
- $Q \approx 1$: graph with strong community structure.

How many clusters in the graph?

- 1 cluster containing all vertices $\rightarrow Q = 0$;
- n clusters, with 1 vertex for cluster $\rightarrow Q < 0$;
- in general $1 < N_c < n$, not known a priori.

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

Modularity maximization is NP-hard `[Brandes et al.; IEEE TKDE, 2008]`.

Modularity

Ranges of modularity

- $-\frac{1}{2} \leq Q \leq 1$ ($-\frac{1}{2}$ for bipartite graphs, 1 cluster for partition);
- $Q \approx 0$: graph similar to random graph;
- $Q \approx 1$: graph with strong community structure.

How many clusters in the graph?

- 1 cluster containing all vertices $\rightarrow Q = 0$;
- n clusters, with 1 vertex for cluster $\rightarrow Q < 0$;
- in general $1 < N_c < n$, not known a priori.

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

Modularity maximization is NP-hard `[Brandes et al.; IEEE TKDE, 2008]`.

Modularity maximization

Exact methods

- Row generation for the clique partitioning model
  [Grötschel, Wakabayashi, Math. Prog. 1989; Brandes et al., IEEE TKDE 2008]
- 0 – 1 MIQP formulation [Xu et al., Eur. Phys. J. B, 2007]
- Column generation extensions [Cafieri et al., Phys. Rev. E, 2010]

Modularity maximization

Exact methods

- Row generation for the clique partitioning model
  [Grötschel, Wakabayashi, Math. Prog. 1989; Brandes et al., IEEE TKDE 2008]
- 0 – 1 MIQP formulation [Xu et al., Eur. Phys. J. B, 2007]
- Column generation extensions [Cafieri et al., Phys. Rev. E, 2010]

Heuristics

- Partitioning methods
  (greedy, simulated annealing, genetic search, and many others)
- Hierarchical divisive and agglomerative methods

For a survey see [Fortunato; Phys. Rep., 2010]

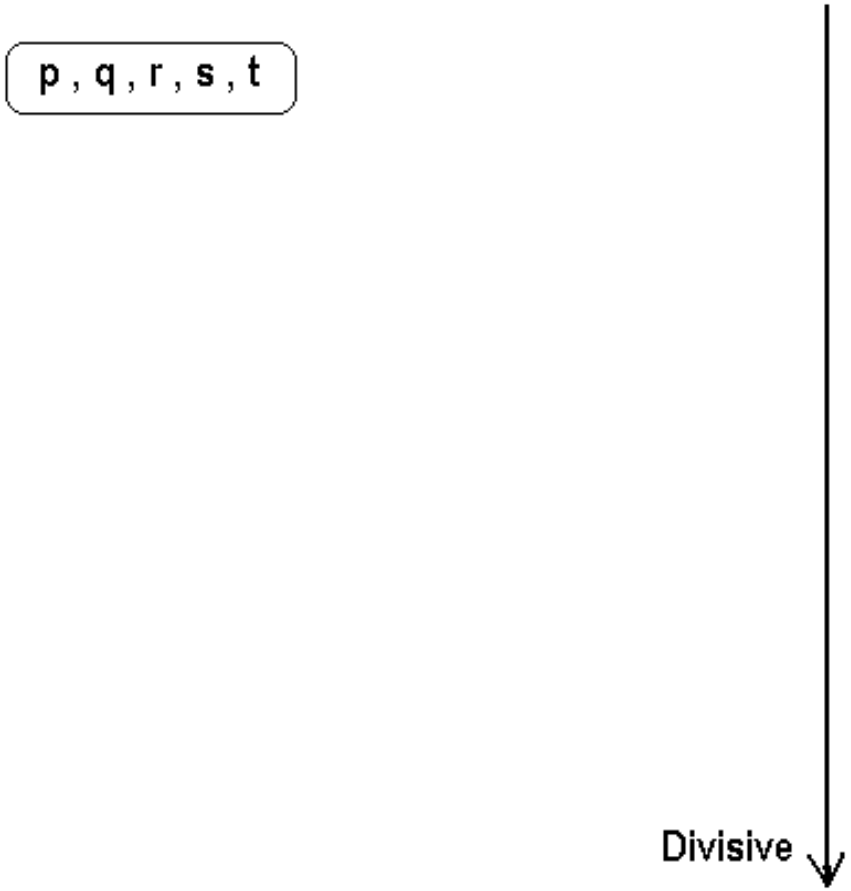
Outline

Hierarchical divisive heuristic

- 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)

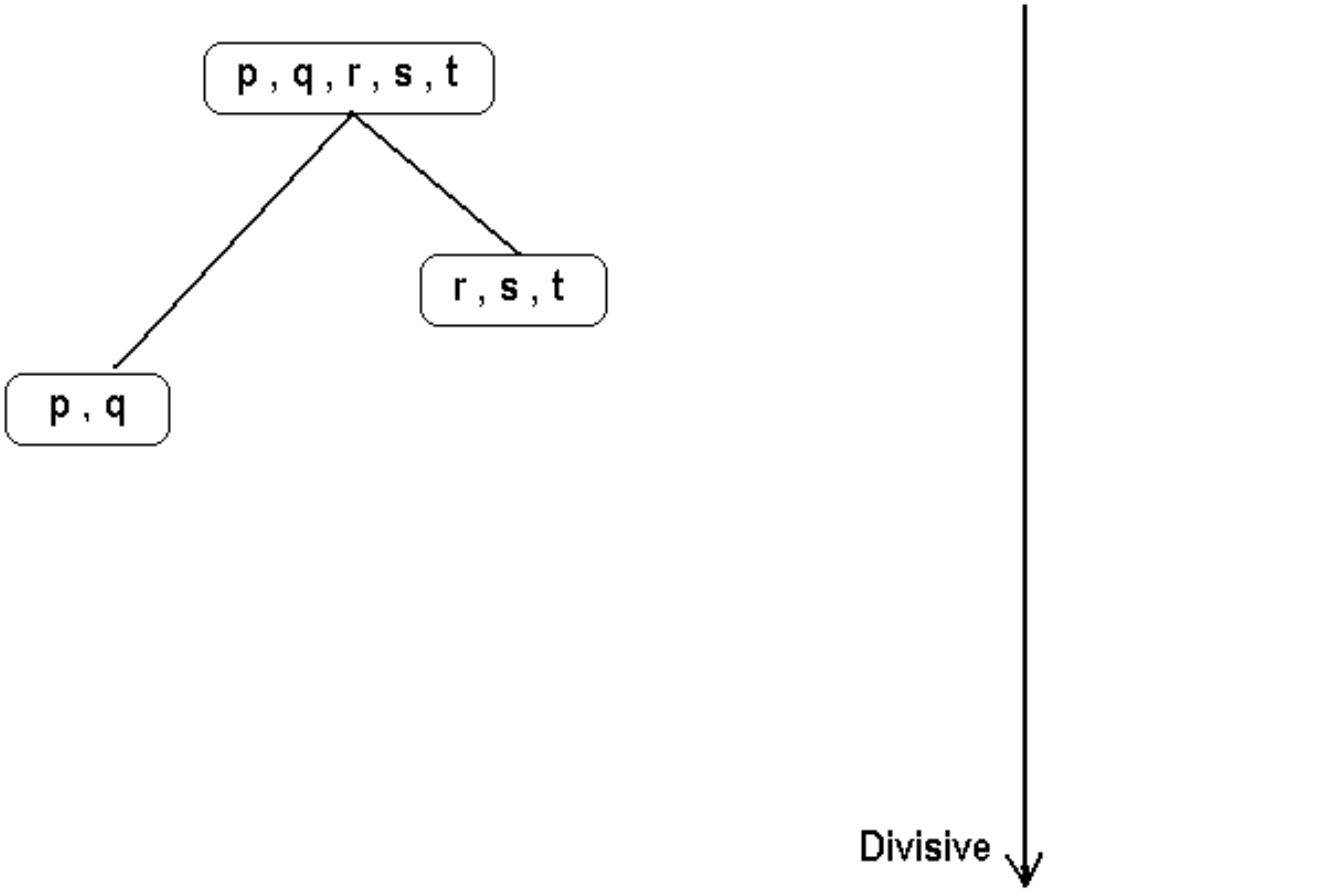
Hierarchical divisive heuristic

- 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)



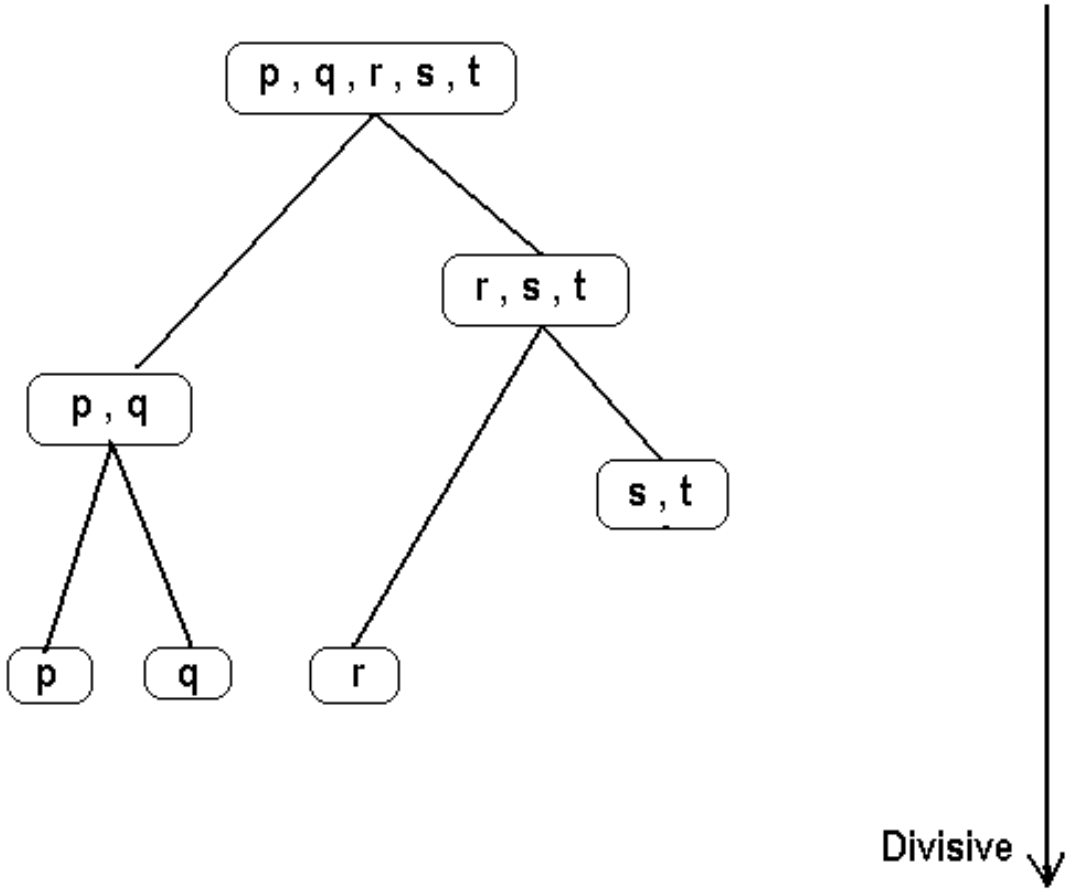
Hierarchical divisive heuristic

- 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)



Hierarchical divisive heuristic

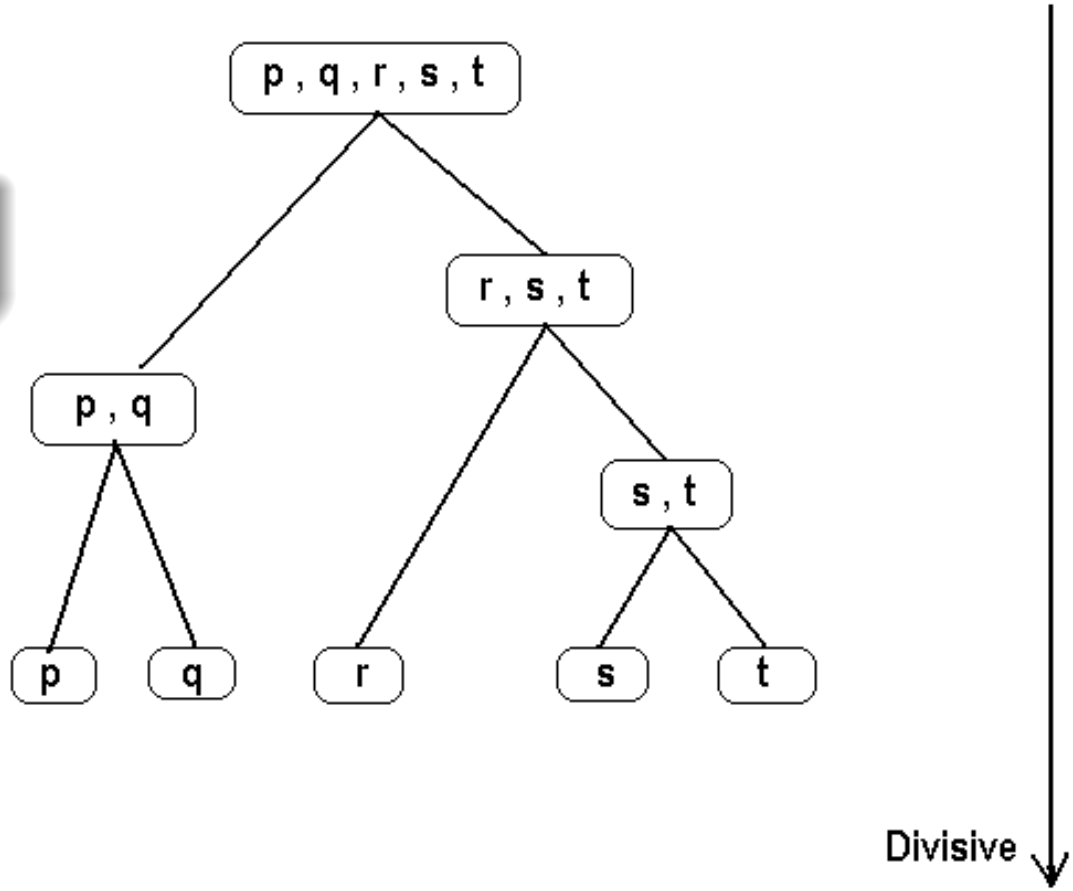
- Proceed from an initial partition containing all entities
- Iteratively divide a cluster into two in such a way to increase most the objective function (or the decrease in the objective value is the smallest possible)



Hierarchical divisive heuristic

- 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 divisive heuristic

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G

Hierarchical divisive heuristic

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G
- Output: partition into clusters of c

Hierarchical divisive heuristic

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G
- Output: partition into clusters of c
- if $|V_c| < 3$ save c as cluster, and return;*

Hierarchical divisive heuristic

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G
- Output: partition into clusters of c
- if $|V_c| < 3$ save c as cluster, and return;*
- divide c in c_1 and c_2

Hierarchical divisive heuristic

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G
- Output: partition into clusters of c
- if $|V_c| < 3$ save c as cluster, and return;*
- divide c in c_1 and c_2
- if $Q(c) > Q(c_1) + Q(c_2)$ save c as cluster, and return;

Hierarchical divisive heuristic

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G
- Output: partition into clusters of c
- if $|V_c| < 3$ save c as cluster, and return;*
- divide c in c_1 and c_2
- if $Q(c) > Q(c_1) + Q(c_2)$ save c as cluster, and return;
- call Divisive(c_1) and Divisive(c_2);

Hierarchical divisive heuristic

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G
- Output: partition into clusters of c
- if $|V_c| < 3$ save c as cluster, and return;$*$
- divide c in c_1 and c_2
- if $Q(c) > Q(c_1) + Q(c_2)$ save c as cluster, and return;
- call Divisive(c_1) and Divisive(c_2);

$*$: an optimal solution does not contain single vertex clusters [Brandes et al.; IEEE TKDE, 2008].

Locally optimal hierarchical divisive heuristic

[Cafieri, Hansen, Liberti.; Phys. Rev. E, 2011]:

- Divisive scheme
- Bipartitioning step modeled as a 0-1 MIQP problem
- The 0-1 MIQP problem exactly solved

$$\Downarrow$$

the proposed heuristic is *locally optimal*

(but not globally optimal)

MIQP model for bipartition in the divisive heuristic

Objective function

Express D_2 as a function of D_1: $D_2 = D_c - D_1$

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

$\Rightarrow$ Modularity:

$$
\begin{aligned}
Q &= \left(\frac{m_1 + m_2}{m} - \frac{{D_1}^2 + {D_2}^2}{4m^2} \right) = \\
&= \frac{m_1 + m_2}{m} - \frac{D_1^2}{4m^2} - \frac{D_c^2 + D_1^2 - 2D_1 D_c}{4m^2} = \\
&= \frac{m_1 + m_2}{m} - \frac{D_1^2}{4m^2} - \frac{D_c^2}{4m^2} + \frac{D_1 D_c}{2m^2}
\end{aligned}
$$

MIQP model for bipartition in the divisive heuristic

Objective function

Express D_2 as a function of D_1: $D_2 = D_c - D_1$

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

$\Rightarrow$ Modularity:

$$
\begin{aligned}
Q &= \left(\frac{m_1 + m_2}{m} - \frac{{D_1}^2 + {D_2}^2}{4m^2} \right) = \\
&= \frac{m_1 + m_2}{m} - \frac{D_1^2}{4m^2} - \frac{D_c^2 + D_1^2 - 2D_1 D_c}{4m^2} = \\
&= \frac{m_1 + m_2}{m} - \frac{D_1^2}{4m^2} - \frac{D_c^2}{4m^2} + \frac{D_1 D_c}{2m^2}
\end{aligned}
$$

Variables

- $X_{i,j,s} = 1$ if the edge (v_i, v_j) is inside the cluster s, 0 otherwise (s is either 1 or 2)
- $Y_i = 1$ if the vertex v_i is inside the cluster 1, 0 otherwise

MIQP model for bipartition in the divisive heuristic

Constraints

Any edge (v_i, v_j) with end vertices indiced by i and j can only belong to cluster s if both of its end vertices also belong to that cluster:

$$\forall (v_i, v_j) \in E_c \qquad X_{i,j,1} \leq Y_{i,1}$$
$$\forall (v_i, v_j) \in E_c \qquad X_{i,j,1} \leq Y_{j,1}$$
$$\forall (v_i, v_j) \in E_c \qquad X_{i,j,2} \leq 1 - Y_{i,1}$$
$$\forall (v_i, v_j) \in E_c \qquad X_{i,j,2} \leq 1 - Y_{j,1}$$

The number of edges of each of the two clusters and the sum of vertex degrees of the first cluster are expressed in terms of X and Y:

$$\forall s \in \{1, 2\} \qquad m_s = \sum_{(v_i, v_j) \in E_c} X_{i,j,s}$$

$$D_1 = \sum_{v_i \in V_c} k_i Y_{i,1}$$

k_i = degree of the vertex v_i

V_c and E_c = respectively the set of vertices and the set of edges of the cluster c to be bipartitioned

MIQP model used by the hierarchical divisive heuristic (*OB* model)

$$\max \quad \frac{1}{m}\left(m_1 + m_2 - \frac{1}{2m}\left({D_1}^2 + \frac{{D_c}^2}{2} - D_1 D_c\right)\right)$$

$$\text{s.t.} \quad X_{i,j,1} \leq Y_i \qquad \forall (v_i, v_j) \in E_c$$

$$X_{i,j,1} \leq Y_j \qquad \forall (v_i, v_j) \in E_c$$

$$X_{i,j,2} \leq 1 - Y_i \qquad \forall (v_i, v_j) \in E_c$$

$$X_{i,j,2} \leq 1 - Y_j \qquad \forall (v_i, v_j) \in E_c$$

$$m_s = \sum_{(v_i, v_j) \in E_c} X_{i,j,s} \qquad \forall s \in \{1, 2\}$$

$$D_1 = \sum_{v_i \in V_c} k_i Y_i$$

$$Y_i \in \{0, 1\} \qquad \forall v_i \in V_c$$

$$X_{i,j,s} \geq 0 \qquad \forall (v_i, v_j) \in E_c, \ \forall s \in \{1, 2\}$$

Outline

Reformulations in Mathematical Programming

$$\left.\begin{array}{r} \min f(x) \\ g(x) \leq b \\ x \in X \end{array}\right\} \Rightarrow$$

Mathematical programming formulation P = a set of entities:

- parameters
- decision variables ($x \in X$)
- objective function ($f(x)$)
- constraints ($g(x) \leq b$)

A *reformulation* Q of P is obtained via (symbolic) transformations applied to x, $f(x)$, $g(x) \leq 0$, whilst keeping some of the properties of P invariant.

P and Q

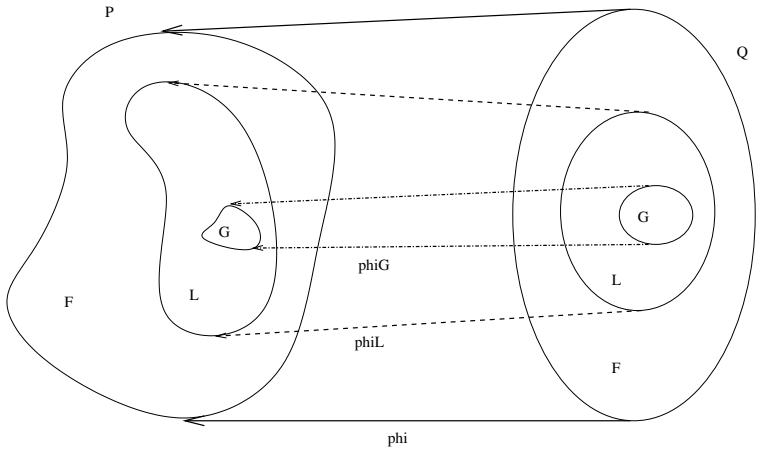
- ⋆ may share the same numerical properties (feasible region, optima)
- ⋆ often perform differently according to the type of the solving algorithm

$\Rightarrow$ choosing the best possible formulation is crucial

Reformulations in Mathematical Programming

Four main reformulation classes (Liberti, 2010):

- *exact reformulations*: preserve all optimality properties
- *narrowings*: preserve some optimality properties
- *relaxations*: provide bounds to the optimal value of the objective function
- *approximations*: formulations $Q(k)$ parametrized on k such that $\lim_{k\to\infty} Q(k)$ is an exact reformulation



Exact reformulation

Improving the 0 – 1 MIQP formulation for bipartition

Reformulations:

- Reduction of number of variables and constraints
- Binary decomposition for linearization of quadratic term D_1^2 in the objective function
- Symmetry breaking

Outline

Reduction of the number of variables

Consider the variables X of the original model:

$$X_{i,j,s} = \begin{cases} 1 & \text{if edge } (v_i, v_j) \text{ belongs to cluster } s \\ 0 & \text{otherwise} \end{cases}$$

Reduction of the number of variables

Consider the variables X of the original model:

$$X_{i,j,s} = \begin{cases} 1 & \text{if edge } (v_i, v_j) \text{ belongs to cluster } s \\ 0 & \text{otherwise} \end{cases}$$

We do not actually need to know if an edge is in the cluster 1 or 2, but only if it is within a cluster or not:

$$X_{i,j} = \begin{cases} 1 & \text{if } Y_i = Y_j \\ 0 & \text{otherwise} \end{cases}$$

$\Rightarrow$ Half of the variables X needed

Reduction of the number of variables

$X_{i,j}$ can be seen as the negation of the XOR operation between Y_i and Y_j variables.

$\Rightarrow$ the following constraints can be employed [Brown and Dell, 2007]

$$\forall (v_i, v_j) \in E_c \qquad X_{i,j} \leq Y_i - Y_j + 1$$

$$\forall (v_i, v_j) \in E_c \qquad X_{i,j} \leq Y_j - Y_i + 1$$

$$\forall (v_i, v_j) \in E_c \qquad X_{i,j} \geq Y_i + Y_j - 1$$

$$\forall (v_i, v_j) \in E_c \qquad X_{i,j} \geq 1 - Y_i - Y_j$$

New variables

Variables X can then be expressed as

$$X_{i,j} = 2Y_iY_j - Y_i - Y_j + 1, \qquad \forall (v_i, v_j) \in E_c$$

New variables

Variables X can then be expressed as

$$X_{i,j} = 2Y_iY_j - Y_i - Y_j + 1, \qquad \forall (v_i, v_j) \in E_c$$

Variables S linearize the product of the binary variables Y:

$$S_{i,j} = Y_iY_j \qquad \forall (v_i, v_j) \in E_c$$

So we obtain

$$X_{i,j} = 2S_{i,j} - Y_i - Y_j + 1 \qquad \forall (v_i, v_j) \in E_c$$

Fortet's linearization

Relationship $S_{i,j} = Y_i Y_j$ (Fortet's inequalities):

$$
\begin{aligned}
S_{i,j} &\geq 0 && \forall (v_i, v_j) \in E_c \\
S_{i,j} &\geq Y_j + Y_i - 1 && \forall (v_i, v_j) \in E_c \\
S_{i,j} &\leq Y_i && \forall (v_i, v_j) \in E_c \\
S_{i,j} &\leq Y_j && \forall (v_i, v_j) \in E_c
\end{aligned}
$$

exact linearization of a product of binary variables

Fortet's linearization

Relationship $S_{i,j} = Y_i Y_j$ (Fortet's inequalities):

$$
\begin{aligned}
S_{i,j} &\geq 0 && \forall (v_i, v_j) \in E_c \\
S_{i,j} &\geq Y_j + Y_i - 1 && \forall (v_i, v_j) \in E_c \\
S_{i,j} &\leq Y_i && \forall (v_i, v_j) \in E_c \\
S_{i,j} &\leq Y_j && \forall (v_i, v_j) \in E_c
\end{aligned}
$$

exact linearization of a product of binary variables

Objective function maximizes variables $S \rightarrow$ half of the constraints needed:

$$
\begin{aligned}
S_{i,j} &\leq Y_i && \forall (v_i, v_j) \in E_c \\
S_{i,j} &\leq Y_j && \forall (v_i, v_j) \in E_c
\end{aligned}
$$

OB_1 formulation

$$
\begin{aligned}
\max \quad & \frac{1}{m}\left(\sum_{(v_i,v_j)\in E_c}\left(2S_{i,j}-Y_i-Y_j\right)+|E_c|-\frac{1}{2m}\left({D_1}^2+\frac{{D_c}^2}{2}-D_1D_c\right)\right) \\
\text{s.t.} \quad & S_{i,j}\leq Y_i && \forall (v_i,v_j)\in E_c \\
& S_{i,j}\leq Y_j && \forall (v_i,v_j)\in E_c \\
& D_1=\sum_{v_i\in V_c} k_iY_i \\
& Y_i\in\{0,1\} && \forall v_i\in V_c
\end{aligned}
$$

where in the objective function we use the fact that $\sum_{(v_i,v_j)\in E_C} 1 = |E_c|$

Outline

Quadratic term reformulation

In the objective function it appears D_1^2, where $D_1 = \sum_{v_i \in V_c} k_i Y_i$.

Quadratic term reformulation

In the objective function it appears D_1^2, where $D_1 = \sum_{v_i \in V_c} k_i Y_i$.

The model is not linear; how to solve the problem?

- using general MINLP solvers (as Couenne or BARON): too time consuming;

Quadratic term reformulation

In the objective function it appears D_1^2, where $D_1 = \sum_{v_i \in V_c} k_i Y_i$.

The model is not linear; how to solve the problem?

- using general MINLP solvers (as Couenne or BARON): too time consuming;
- using convex quadratic solver as CPLEX: efficient, but not possible to employ other MIP solvers;

Quadratic term reformulation

In the objective function it appears D_1^2, where $D_1 = \sum_{v_i \in V_c} k_i Y_i$.

The model is not linear; how to solve the problem?

- using general MINLP solvers (as Couenne or BARON): too time consuming;
- using convex quadratic solver as CPLEX: efficient, but not possible to employ other MIP solvers;
- linearize products $Y_i Y_j$ using Fortet's inequalities: many variables and constraints;

Quadratic term reformulation

In the objective function it appears D_1^2, where $D_1 = \sum_{v_i \in V_c} k_i Y_i$.

The model is not linear; how to solve the problem?

- using general MINLP solvers (as Couenne or BARON): too time consuming;
- using convex quadratic solver as CPLEX: efficient, but not possible to employ other MIP solvers;
- linearize products $Y_i Y_j$ using Fortet's inequalities: many variables and constraints;
- linearization using binary decomposition [Billionnet, Elloumi, Lambert; Math. Prog., 2012]: better than Fortet inequalities.

Binary decomposition reformulation

$$D_1 = \sum_{v_i \in V_c} k_i Y_i$$

k_i is integer (degree of vertex v_i), Y_i is binary $\rightarrow$ D_1 is integer.

Binary decomposition reformulation

$$D_1 = \sum_{v_i \in V_c} k_i Y_i$$

k_i is integer (degree of vertex v_i), Y_i is binary $\rightarrow$ D_1 is integer.

$$D_1 = \sum_{l=0}^{t} 2^l a_l, \quad a_l \in \{0, 1\}$$

where $t = \lceil \log_2(D_c + 1) - 1 \rceil$.

Binary decomposition reformulation

$$D_1 = \sum_{v_i \in V_c} k_i Y_i$$

k_i is integer (degree of vertex v_i), Y_i is binary $\rightarrow$ D_1 is integer.

$$D_1 = \sum_{l=0}^{t} 2^l a_l, \quad a_l \in \{0, 1\}$$

where $t = \lceil \log_2(D_c + 1) - 1 \rceil$.

$$D_1{}^2 = \sum_{l=0}^{t} 2^l a_l \cdot \sum_{h=0}^{t} 2^h a_h = \sum_{l=0}^{t} 2^{2l} a_l + \sum_{l=0}^{t} \sum_{h<l} 2^{l+h+1} a_l a_h$$

Linearization for binary decomposition

$a_l a_h$ product of binary variables, can be linearized

$\Rightarrow$ $R_{l,h}$ linearize the product $a_l a_h$.

$-R$ is maximized in the objective function $\rightarrow$ only 2 constraints are needed:

$$R_{l,h} \geq 0 \qquad \forall l \in \{0, \ldots, t\},\ \forall h \in \{0, \ldots, l-1\}$$

$$R_{l,h} \geq a_l + a_h - 1 \qquad \forall l \in \{0, \ldots, t\},\ \forall h \in \{0, \ldots, l-1\}$$

Linearization for binary decomposition

$a_l a_h$ product of binary variables, can be linearized

$\Rightarrow$ $R_{l,h}$ linearize the product $a_l a_h$.

$-R$ is maximized in the objective function $\rightarrow$ only 2 constraints are needed:

$$R_{l,h} \geq 0 \qquad \forall l \in \{0,\ldots,t\},\ \forall h \in \{0,\ldots,l-1\}$$

$$R_{l,h} \geq a_l + a_h - 1 \qquad \forall l \in \{0,\ldots,t\},\ \forall h \in \{0,\ldots,l-1\}$$

The objective function is

$$\frac{1}{m}\left(m_1 + m_2 - \frac{1}{2m}\left(\sum_{l=0}^{t} 2^{2l} a_l + \sum_{l=0}^{t}\sum_{h<l} 2^{l+h+1} R_{lh} + \frac{{D_c}^2}{2} - D_1 D_c\right)\right)$$

OB_2 formulation

$$\max \quad \frac{1}{m}\left(m_1 + m_2 - \frac{1}{2m}\left(\sum_{l=0}^{t} 2^{2l} a_l + \sum_{l=0}^{t}\sum_{h<l} 2^{l+h+1} R_{lh} + \frac{{D_c}^2}{2} - D_1 D_c\right)\right)$$

$$\text{s.t.} \quad X_{i,j,1} \leq Y_i \qquad \forall (v_i, v_j) \in E_c$$

$$X_{i,j,1} \leq Y_j \qquad \forall (v_i, v_j) \in E_c$$

$$X_{i,j,2} \leq 1 - Y_i \qquad \forall (v_i, v_j) \in E_c$$

$$X_{i,j,2} \leq 1 - Y_j \qquad \forall (v_i, v_j) \in E_c$$

$$R_{l,h} \geq a_l + a_h - 1 \qquad \forall l \in \{0, \ldots, t\},\ \forall h \in \{0, \ldots, l-1\}$$

$$m_s = \sum_{(v_i, v_j) \in E_c} X_{i,j,s} \qquad \forall s \in \{1, 2\}$$

$$D_1 = \sum_{v_i \in V_c} k_i Y_i$$

$$D_1 = \sum_{l=0}^{t} 2^l a_l$$

$$Y_i \in \{0, 1\} \qquad \forall v_i \in V_c$$

$$X_{i,j,s} \geq 0 \qquad \forall (v_i, v_j) \in E_c,\ \forall s \in \{1, 2\}$$

$$R_{l,h} \geq 0 \qquad \forall l \in \{0, \ldots, t\},\ \forall h \in \{0, \ldots, l-1\}$$

Compact binary decomposition 1

It is possible to reduce the number of variables R.

$R_{l,h}$ is the linearization of the term $a_l a_h$.
We can write the term of the objective function involving $R_{l,h}$ in this way:

$$\sum_{l=0}^{t} \sum_{h<l} 2^{l+h+1} R_{lh} = \sum_{l=0}^{t} \sum_{h<l} 2^{l+h+1} a_l a_h = \sum_{l=0}^{t} 2^{l+1} a_l \sum_{h<l} 2^h a_h = \sum_{l=0}^{t} 2^{l+1} a_l b_l = \sum_{l=0}^{t} 2^{l+1} R_l,$$

where $R_l = a_l b_l$ and b_l is a new variable defined as $\sum_{h<l} 2^h a_h$.
Since the upper bound for b_l is $U_{b_l} = \sum_{h<l} 2^h = 2^l - 1$,
the constraints to add to the model are:

$$\forall l \in \{0, \ldots, t\} \quad b_l = \sum_{h<l} 2^h a_h$$

$$\forall l \in \{0, \ldots, t\} \quad R_l \geq 0$$

$$\forall l \in \{0, \ldots, t\} \quad R_l \geq U_{b_l} a_l + b_l - U_{b_l}$$

We have now $t + 1$ variables R_l instead of $\frac{t^2+t}{2}$ variables $R_{l,h}$,
and we have adjoined $t + 1$ variables b and $t + 1$ constraints (definition of b).

Compact binary decomposition 2

We can put together the term containing the variables a_l and R_l:

$$\sum_{l=0}^{t} 2^{2l} a_l + \sum_{l=0}^{t} 2^{l+1} R_l = \sum_{l=0}^{t} 2^{2l} a_l + \frac{2^{2l}}{2^{l-1}} R_l = \sum_{l=0}^{t} 2^{2l} \left(a_l + \frac{a_l b_l}{2^{l-1}} \right).$$

Hence

$$\sum_{l=0}^{t} 2^{2l} \left(a_l + \frac{a_l b_l}{2^{l-1}} \right) = \sum_{l=0}^{t} \frac{2^{2l}}{2^{l-1}} a_l \left(b_l + 2^{l-1} \right) = \sum_{l=0}^{t} 2^{l+1} a_l z_l = \sum_{l=0}^{t} 2^{l+1} T_l, \tag{1}$$

where the new variable z_l is equal to $b_l + 2^{l-1}$ and T_l is the linearization of $a_l z_l$.

$\Rightarrow$ we remove the variables R and b, and all the related constraints, and adjoin the new variables z and T, as well as these constraints:

$$\forall l \in \{0, \ldots, t\} \quad z_l = \sum_{h<l} 2^h a_h + 2^{l-1} \tag{2}$$

$$\forall l \in \{0, \ldots, t\} \quad T_l \geq 0 \tag{3}$$

$$\forall l \in \{0, \ldots, t\} \quad T_l \geq U_{z_l} a_l + z_l - U_{z_l}, \tag{4}$$

where U_{z_l} is the upper bound of the variable z_l, and it is equal to 2^l.

Outline

Symmetry breaking constraint - Fixing a vertex

If a solution is found, another equivalent solution is obtained by swapping the clusters (i.e., vertices in cluster 1 are placed in cluster 2, and vice-versa)

$\rightarrow$ fix a vertex in one of the clusters

Symmetry breaking constraint - Fixing a vertex

If a solution is found, another equivalent solution is obtained by swapping the clusters (i.e., vertices in cluster 1 are placed in cluster 2, and vice-versa)

$\rightarrow$ fix a vertex in one of the clusters

Good choice: fix the vertex with highest degree in one cluster

$$Y_g = 0, \qquad g = \arg\max\{k_i, \ \forall v_i \in V_c\}$$

Symmetry breaking constraint - Fixing a vertex

If a solution is found, another equivalent solution is obtained by swapping the clusters (i.e., vertices in cluster 1 are placed in cluster 2, and vice-versa)

$\rightarrow$ fix a vertex in one of the clusters

Good choice: fix the vertex with highest degree in one cluster

$$Y_g = 0, \qquad g = \arg\max\{k_i, \forall v_i \in V_c\}$$

Original formulation + symmetry breaking constraint $\rightarrow OB_3$ model.

Outline

Tests

- original formulation (OB)

Tests

- original formulation (OB)
- reformulation with less variables and constraints (OB_1)

Tests

- original formulation (OB)
- reformulation with less variables and constraints (OB_1)
- binary decomposition reformulation (OB_2)

Tests

- original formulation (OB)
- reformulation with less variables and constraints (OB_1)
- binary decomposition reformulation (OB_2)
- original formulation + symmetry breaking constraint (OB_3)

Tests

- original formulation (OB)
- reformulation with less variables and constraints (OB_1)
- binary decomposition reformulation (OB_2)
- original formulation + symmetry breaking constraint (OB_3)

ID	Graph	n	m	Reference
1	Karate	34	78	Zachary (1977)
2	Dolphins	62	159	Lusseau et al. (2003)
3	*Les misérables*	77	254	Hugo (1951), Knuth (1993)
4	A00 main	83	135	Batagelj and Mrvar (2006)
5	P53 protein	104	226	Dartnell at al. (2005)
6	Political books	105	441	Krebs (2008)
7	Football	115	613	Girvan and Newman (2002)
8	A01 main	249	635	Batagelj and Mrvar (2006)
9	USAir97	332	2126	Batagelj and Mrvar (2006)
10	Netscience main	379	914	Newman (2006a)
11	S838	512	819	Milo et al. (2004)
12	Power	4941	6594	Watts and Strogatz (1998)

Numerical results

Tests: 2.8GHz Intel iCore i7 CPU, 8 GB RAM, Linux, CPLEX 12.2

- branching based on pseudo reduced costs

Numerical results

Tests: 2.8GHz Intel iCore i7 CPU, 8 GB RAM, Linux, CPLEX 12.2
- branching based on pseudo reduced costs

graph			clusters		OB		OB_1		OB_2		OB_3	
	$\|V\|$	$\|E\|$	N_c	Q	nodes	CPU time	nodes	CPU time	nodes	CPU time	nodes	CPU time
1	34	78	4	0.4188	45	0.14	41	**0.06**	123	0.52	**18**	0.07
2	62	159	4	0.5265	207	0.59	157	**0.19**	505	1.29	**98**	0.49
3	77	254	8	0.5468	205	1.09	185	**0.40**	577	2.16	**102**	0.58
4	83	135	7	0.5281	76	0.35	56	0.11	251	0.74	**27**	**0.08**
5	104	226	7	0.5284	275	1.10	201	**0.53**	678	3.22	**135**	0.59
6	105	441	4	0.5263	313	3.04	294	**1.00**	1284	9.17	**145**	1.36
7	115	613	10	0.6009	8853	307.56	5410	**56.69**	15406	252.96	**3014**	118.24
8	249	635	15	0.6288	1119	47.83	1010	**16.85**	4395	61.49	**997**	45.85
9	332	2126	8	0.3596	16682	4585.04	17811	**1041.89**	63687	3074.09	**9446**	2510.81
10	379	914	20	0.8470	291	3.64	267	**1.44**	931	14.53	**108**	1.82
11	512	819	15	0.8166	392	5.26	304	**1.26**	1348	22.46	**197**	2.15
12	4941	6594	4	0.9396	1459	708.51	1449	**217.61**	11289	2029.63	**815**	417.26

Best formulation

Best results: formulation with less constraints and variables (OB_1)
+ symmetry breaking constraint (OB_3)

graph			OB		$OB_1 + OB_3$	
	vertices	edges	nodes	CPU time	nodes	CPU time
Karate	34	78	45	0.14	**17**	**0.04**
Dolphins	62	159	207	0.59	**93**	**0.16**
Les Misèrables	77	254	205	1.09	**105**	**0.35**
A00 main	83	135	76	0.35	**26**	**0.04**
P53 protein	104	226	275	1.10	**119**	**0.26**
Political books	105	441	313	3.04	**152**	**0.51**
Football	115	613	8853	307.56	**3822**	**44.38**
A01 main	249	635	1119	47.83	**726**	**9.72**
USAir97	332	2126	16682	4585.04	**8665**	**446.06**
Netscience main	379	914	291	3.64	**94**	**0.85**
S838	512	819	392	5.26	**186**	**1.18**
Power	4941	6594	1459	708.51	**891**	**123.85**

Outline

Conclusion

Pros

- Reformulations reduce computational time (up to 10 times).

Conclusion

Pros

- Reformulations reduce computational time (up to 10 times).
- High impact of the simple symmetry breaking constraint.

Conclusion

Pros

- Reformulations reduce computational time (up to 10 times).
- High impact of the simple symmetry breaking constraint.

Cons

- Binary decomposition not very effective (it is better to use CPLEX with the quadratic model).

Conclusion

Pros

- Reformulations reduce computational time (up to 10 times).
- High impact of the simple symmetry breaking constraint.

Cons

- Binary decomposition not very effective (it is better to use CPLEX with the quadratic model).

Future work

- Improve the binary decomposition.
- Use the model for divisive heuristic for bipartite graph (there is not a quadratic term there, binary decomposition is the best choice).