Binary optimization models for computing the frustration index in signed graphs

Mark C. Wilson
(joint with Samin Aref & Andrew J. Mason)

Department of Computer Science
University of Auckland
www.cs.auckland.ac.nz/~mcw/

ORSNZ, AUT, 2016-11-29

Balance theory

- In social networks, if A, B, C are mutually related, and A and B are friends, A and C are friends, but B and C are not friends, there is social tension.

Balance theory

- In social networks, if A, B, C are mutually related, and A and B are friends, A and C are friends, but B and C are not friends, there is social tension.
- Heider (1940s) postulated that such situations tend to become balanced, so that two friends have a common enemy, or all three become friends.

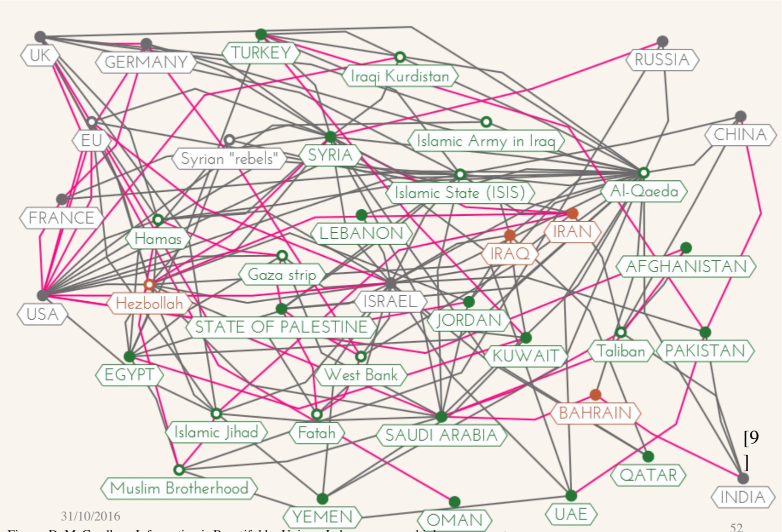
Balance theory

- In social networks, if A, B, C are mutually related, and A and B are friends, A and C are friends, but B and C are not friends, there is social tension.
- Heider (1940s) postulated that such situations tend to become balanced, so that two friends have a common enemy, or all three become friends.
- This idea of balance in a network has been used in statistical physics (spin glass models), biology, finance, knot theory, coding theory, international relations (alliance and enmity), chemistry, materials science, electronics.

Balance theory

- In social networks, if A, B, C are mutually related, and A and B are friends, A and C are friends, but B and C are not friends, there is social tension.
- Heider (1940s) postulated that such situations tend to become balanced, so that two friends have a common enemy, or all three become friends.
- This idea of balance in a network has been used in statistical physics (spin glass models), biology, finance, knot theory, coding theory, international relations (alliance and enmity), chemistry, materials science, electronics.
- Despite many papers on this topic, there is no standard measure of partial balance. In previous work we axiomatized various measures of balance and recommended the *frustration index*, arising in physics.

Middle East signed network



[9
]

Figure: D. McCandless, Information is Beautiful by Univers Labs, source: multiple news reports

The basic graph problem

- Consider a finite undirected graph $G = (V, E)$ equipped with an edge weight function $\sigma : E \to \{\pm 1\}$. This is a signed graph, with positive edges E^+ and negative edges E^-.

The basic graph problem

- Consider a finite undirected graph $G = (V, E)$ equipped with an edge weight function $\sigma : E \to \{\pm 1\}$. This is a signed graph, with positive edges E^+ and negative edges E^-.
- Let $m^- = |E^-|$ denote the number of negative edges, and $A = (a_{ij})$ the signed adjacency matrix (each entry is ± 1).

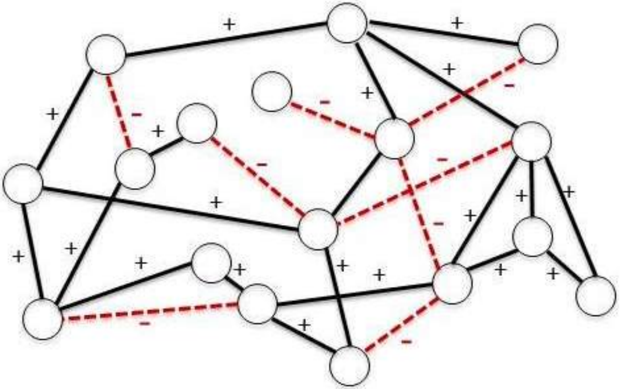
The basic graph problem

- Consider a finite undirected graph $G = (V, E)$ equipped with an edge weight function $\sigma : E \to \{\pm 1\}$. This is a signed graph, with positive edges E^+ and negative edges E^-.
- Let $m^- = |E^-|$ denote the number of negative edges, and $A = (a_{ij})$ the signed adjacency matrix (each entry is ± 1).
- If we can colour each vertex 0 or 1, then a positive edge is frustrated if its endpoints have different colours, while a negative edge is frustrated if its endpoints have the same colour. A graph is balanced if it has a colouring with no frustration.

The basic graph problem

- Consider a finite undirected graph $G = (V, E)$ equipped with an edge weight function $\sigma : E \to \{\pm 1\}$. This is a signed graph, with positive edges E^+ and negative edges E^-.
- Let $m^- = |E^-|$ denote the number of negative edges, and $A = (a_{ij})$ the signed adjacency matrix (each entry is ± 1).
- If we can colour each vertex 0 or 1, then a positive edge is frustrated if its endpoints have different colours, while a negative edge is frustrated if its endpoints have the same colour. A graph is balanced if it has a colouring with no frustration.
- We aim to compute the frustration index, the minimum number of frustrated edges over all colourings. Alternatively, the minimum number of edges which must be negated (or deleted) in order to make the graph balanced.

Example — signed graph



Previous work

- Determining whether the frustration index is less than k is NP-complete by reduction from MAX-CUT. It can be solved in polynomial time for planar graphs.

Previous work

- Determining whether the frustration index is less than k is NP-complete by reduction from MAX-CUT. It can be solved in polynomial time for planar graphs.
- Many authors have discussed heuristic methods, based on local search for example.

Previous work

- Determining whether the frustration index is less than k is NP-complete by reduction from MAX-CUT. It can be solved in polynomial time for planar graphs.
- Many authors have discussed heuristic methods, based on local search for example.
- Some authors have presented polynomial-time approximation algorithms.

Previous work

- Determining whether the frustration index is less than k is NP-complete by reduction from MAX-CUT. It can be solved in polynomial time for planar graphs.
- Many authors have discussed heuristic methods, based on local search for example.
- Some authors have presented polynomial-time approximation algorithms.
- Apparently, *no one has seriously attempted to solve the problem exactly on decent sized graphs*. (!)

Previous work

- Determining whether the frustration index is less than k is NP-complete by reduction from MAX-CUT. It can be solved in polynomial time for planar graphs.
- Many authors have discussed heuristic methods, based on local search for example.
- Some authors have presented polynomial-time approximation algorithms.
- Apparently, *no one has seriously attempted to solve the problem exactly on decent sized graphs*. (!)
- This is precisely what we do in the paper. We obtain good performance results, and also find errors in previous work.

Basic model

- For each edge e, define variable f_e to indicate whether the edge is frustrated by a given colouring.

Basic model

- For each edge e, define variable f_e to indicate whether the edge is frustrated by a given colouring.
- The frustration index is then computed by

$$\min_{x} \sum_{e \in E} f_e$$

where $x : V \to \{0, 1\}$ is a colouring.

Basic model

- For each edge e, define variable f_e to indicate whether the edge is frustrated by a given colouring.
- The frustration index is then computed by

$$\min_x \sum_{e \in E} f_e$$

where $x : V \to \{0, 1\}$ is a colouring.

- All our models use this skeleton, with different expressions for f_e, which necessitate different constraints.

Frustrated edges

- An edge is frustrated if and only if it is positive and links nodes with a different colour, or negative and links nodes of the same colour.

Frustrated edges

- An edge is frustrated if and only if it is positive and links nodes with a different colour, or negative and links nodes of the same colour.
- Thus

$$f_{\{u,v\}} = \begin{cases} 0, & \text{if } x_u = x_v \text{ and } (u,v) \in E^+ \\ 1, & \text{if } x_u = x_v \text{ and } (u,v) \in E^- \\ 0, & \text{if } x_u \neq x_v \text{ and } (u,v) \in E^- \\ 1, & \text{if } x_u \neq x_v \text{ and } (u,v) \in E^+ \end{cases}$$

Frustrated edges

- An edge is frustrated if and only if it is positive and links nodes with a different colour, or negative and links nodes of the same colour.
- Thus

$$f_{\{u,v\}} = \begin{cases} 0, & \text{if } x_u = x_v \text{ and } (u,v) \in E^+ \\ 1, & \text{if } x_u = x_v \text{ and } (u,v) \in E^- \\ 0, & \text{if } x_u \neq x_v \text{ and } (u,v) \in E^- \\ 1, & \text{if } x_u \neq x_v \text{ and } (u,v) \in E^+ \end{cases}$$

- Note that for a fixed colouring, $f_e = 1 - f(-e)$, where $-e$ denotes the negated edge, so we only need to specify the formulae below for $e \in E^+$.

First model - quadratic

Here we represent f directly as follows, with $e = \{u, v\} \in E^+$.

$$f_e = \text{ XOR } (x_u, x_v) = (x_u \wedge \neg x_v) \vee (x_v \wedge \neg x_u).$$

Replacing each edge by two *directed* edges of the form $e' = (u, v)$ we can consider the simpler formula $f_{e'} = x_u \wedge \neg x_v = x_u - x_u x_v$. This gives a quadratic 0-1 integer programming model.

$$\begin{aligned} \min & \sum_{(u,v)\in E^+} x_u(1 - x_v) + \sum_{(u,v)\in E^-} 1 - x_u(1 - x_v) \\ s/t \quad & x_u \in \{0, 1\}, u \in V \end{aligned}$$

Second model - AND

We simply replace $x_u x_v$ with a new variable x_{uv} and include extra constraints to enforce this equality.

$$\min \sum_{(u,v)\in E^+} x_u - x_{uv} + \sum_{(u,v)\in E^-} 1 - x_u + x_{uv}$$

$$\begin{aligned} x_{uv} &\leq (x_u + x_v)/2 \quad (u,v) \in E^+ \\ x_{uv} &\geq x_u + x_v - 1 \quad (u,v) \in E^- \\ x_u, x_{uv} &\in \{0,1\} \end{aligned}$$

We use the pressure of the objective function to remove non-binding constraints.

Third model - XOR

We make the change of variable $w_{uv} = \text{XOR}(x_u, x_v)$ and enforce this with extra constraints.

$$\min \sum_{\{u,v\} \in E^+} w_{uv} + \sum_{\{u,v\} \in E^-} 1 - w_{uv}$$

$$\begin{aligned}
w_{uv} &\geq x_u - x_v \quad \{u,v\} \in E^+ \\
w_{uv} &\geq x_v - x_u \quad \{u,v\} \in E^+ \\
w_{uv} &\leq 2 - x_u - x_v \quad \{u,v\} \in E^- \\
w_{uv} &\leq x_u + x_v \quad \{u,v\} \in E^- \\
x_u, w_{uv} &\in \{0,1\}
\end{aligned}$$

Fourth model - ABS

We instead express f via $f_{\{u,v\}} = |x_u - x_v|$ for $\{u,v\} \in E^+$. With $2a_{uv} = |x_u - x_v| + (x_u - x_v)$, $2b_{uv} = |x_u - x_v| - (x_u - x_v)$ we obtain

$$
\begin{aligned}
\min \sum_{uv \in E} a_{uv} + b_{uv}& \\
x_u - x_v = a_{uv} - b_{uv} &\quad \{u,v\} \in E^+ \\
x_u + x_v - 1 = a_{uv} - b_{uv} &\quad \{u,v\} \in E^- \\
x_u, a_{uv}, b_{uv} \in \{0,1\}&
\end{aligned}
$$

Summary of models

Table: Comparison of the variables and constraints in the models

	First	Second	Third	Fourth
Variables	n	$n+m$	$n+m$	$n+2m$
Constraints	0	m	$2m$	m
Variable type	binary	binary	binary	binary
Constraint type	-	linear$\leq$	linear$\leq$	linear$=$
Objective	quadratic	linear	linear	linear

Additional details

- There are several conditions that must be satisfied by feasible solutions.

Additional details

- There are several conditions that must be satisfied by feasible solutions.
- We implement them in Gurobi using lazy constraints. Upon violation by a solution, lazy constraints are efficiently pulled into the model in order to cut a part of the feasible space.

Additional details

- There are several conditions that must be satisfied by feasible solutions.
- We implement them in Gurobi using lazy constraints. Upon violation by a solution, lazy constraints are efficiently pulled into the model in order to cut a part of the feasible space.
- The various models incorporate known valid inequalities such as:

Additional details

- There are several conditions that must be satisfied by feasible solutions.
- We implement them in Gurobi using lazy constraints. Upon violation by a solution, lazy constraints are efficiently pulled into the model in order to cut a part of the feasible space.
- The various models incorporate known valid inequalities such as:
  - in the optimal solution, the signed degree of each vertex is nonnegative;

Additional details

- There are several conditions that must be satisfied by feasible solutions.
- We implement them in Gurobi using lazy constraints. Upon violation by a solution, lazy constraints are efficiently pulled into the model in order to cut a part of the feasible space.
- The various models incorporate known valid inequalities such as:
  - in the optimal solution, the signed degree of each vertex is nonnegative;
  - every unbalanced cycle of the graph contains an odd number of frustrated edges, so in particular $\sum_e f_e \geq 1$ for every unbalanced triangle.

Additional details

- There are several conditions that must be satisfied by feasible solutions.
- We implement them in Gurobi using lazy constraints. Upon violation by a solution, lazy constraints are efficiently pulled into the model in order to cut a part of the feasible space.
- The various models incorporate known valid inequalities such as:
  - in the optimal solution, the signed degree of each vertex is nonnegative;
  - every unbalanced cycle of the graph contains an odd number of frustrated edges, so in particular $\sum_e f_e \geq 1$ for every unbalanced triangle.
- We also break symmetry by colouring the root node 1, and use heuristics for the branching priority (decreasing order of unsigned degree).

Test problems

- We used some standard social network examples: Read's dataset for New Guinean highland tribes (G1); Sampson's dataset for monastery interactions (G2); graphs inferred from datasets of students' choice and rejection by Newcomb and Lemann (G3 and G4). Network G5 is inferred by Neal through implementing a stochastic degree sequence model on Fowler's data on Senate bill co-sponsorship.

Test problems

- We used some standard social network examples: Read's dataset for New Guinean highland tribes (G1); Sampson's dataset for monastery interactions (G2); graphs inferred from datasets of students' choice and rejection by Newcomb and Lemann (G3 and G4). Network G5 is inferred by Neal through implementing a stochastic degree sequence model on Fowler's data on Senate bill co-sponsorship.
- We used four examples from biology: Epidermal growth factor receptor pathway (G6); represents the molecular interaction map of a macrophage (G7); gene regulatory networks of the yeast Saccharomyces cerevisiae (G8) and the bacterium (Escherichia coli (G9).

Test problems

- We used some standard social network examples: Read's dataset for New Guinean highland tribes (G1); Sampson's dataset for monastery interactions (G2); graphs inferred from datasets of students' choice and rejection by Newcomb and Lemann (G3 and G4). Network G5 is inferred by Neal through implementing a stochastic degree sequence model on Fowler's data on Senate bill co-sponsorship.
- We used four examples from biology: Epidermal growth factor receptor pathway (G6); represents the molecular interaction map of a macrophage (G7); gene regulatory networks of the yeast Saccharomyces cerevisiae (G8) and the bacterium (Escherichia coli (G9).
- We implemented the models using Gurobi Python interface and a desktop computer with an Intel Corei5 4670 @ 3.40 GHz and 8.00 GB of RAM running 64-bit Microsoft Windows 7.

Results - solve time

Table: Comparison of solve time

	G6	G7	G8	G9
HBN2010	15h	1d	5h	

Results - solve time

Table: Comparison of solve time

	G6	G7	G8	G9
HBN2010	15h	1d	5h	
IRSA2010	few min	few min	few min	few min

Results - solve time

Table: Comparison of solve time

	G6	G7	G8	G9
HBN2010	15h	1d	5h	
IRSA2010	few min	few min	few min	few min
AND	2.7s	2.4s	0.9s	44s

Results - solve time

Table: Comparison of solve time

	G6	G7	G8	G9
HBN2010	15h	1d	5h	
IRSA2010	few min	few min	few min	few min
AND	2.7s	2.4s	0.9s	44s
XOR	6.2s	20s	0.7s	1.6s

Results - solve time

Table: Comparison of solve time

	G6	G7	G8	G9
HBN2010	15h	1d	5h	
IRSA2010	few min	few min	few min	few min
AND	2.7s	2.4s	0.9s	44s
XOR	6.2s	20s	0.7s	1.6s
ABS	0.5s	0.5s	0.3s	1.3s

Results -solution quality

Table: Best solution values found

	G6	G7	G8	G9
optimum	193	332	41	371

Results -solution quality

Table: Best solution values found

	G6	G7	G8	G9
optimum	193	332	41	371
HBN2010	[196, 219], 210	[218,383], 374	[0, 43], 41	

Results -solution quality

Table: Best solution values found

	G6	G7	G8	G9
optimum	193	332	41	371
HBN2010	[196, 219], 210	[218,383], 374	[0, 43], 41	
IRSA2010	[186, 193]	[302, 332]	41	[365, 371]

Results -solution quality

Table: Best solution values found

	G6	G7	G8	G9
optimum	193	332	41	371
HBN2010	[196, 219], 210	[218,383], 374	[0, 43], 41	
IRSA2010	[186, 193]	[302, 332]	41	[365, 371]
AND	193	332	41	371
XOR	193	332	41	371
ABS	193	332	41	371

Results - balance

Graph	n	m	m^-	$L(G)$	$L(G_r) \pm$ SD	Z score
G1	16	58	29	7	14.80 ± 1.25	-6.25
G2	18	49	12	5	10.02 ± 1.22	-4.10
G3	17	40	17	4	8.02 ± 0.88	-4.55
G4	17	36	16	6	7.04 ± 1.00	-1.04
G5	100	2461	1047	331	973.83 ± 9.30	-69.13
G6	329	779	264	193	148.82 ± 5.11	8.65
G7	678	1425	478	332	253.16 ± 6.48	12.16
G8	690	1080	220	41	114.90 ± 5.52	-13.39
G9	1461	3212	1336	371	651.58 ± 6.92	-40.55

Conclusions

- Our algorithms work much better in practice than the previously used heuristics.

Conclusions

- Our algorithms work much better in practice than the previously used heuristics.
- This will allow serious tests of balance theory on dynamically changing graphs.

Conclusions

- Our algorithms work much better in practice than the previously used heuristics.
- This will allow serious tests of balance theory on dynamically changing graphs.
- Note that 7 of the 9 test cases were more balanced than we would expect by randomly assigning signs.

Conclusions

- Our algorithms work much better in practice than the previously used heuristics.
- This will allow serious tests of balance theory on dynamically changing graphs.
- Note that 7 of the 9 test cases were more balanced than we would expect by randomly assigning signs.
- It is still possible to improve performance by better use of structural properties and lazy constraints.