Ahmedabad University

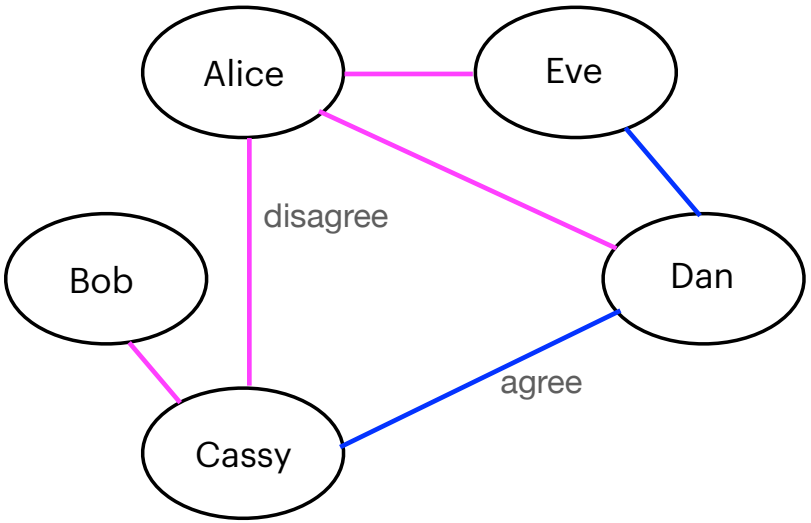
Structural Balance

Amit A. Nanavati
Ahmedabad University

ACM Winter School on Network Science, 2023, Ahmedabad University

Thanks to
Easley, David, and Jon Kleinberg. "Networks, crowds, and markets." Cambridge Books (2012).

A Signed *Opinion* Graph

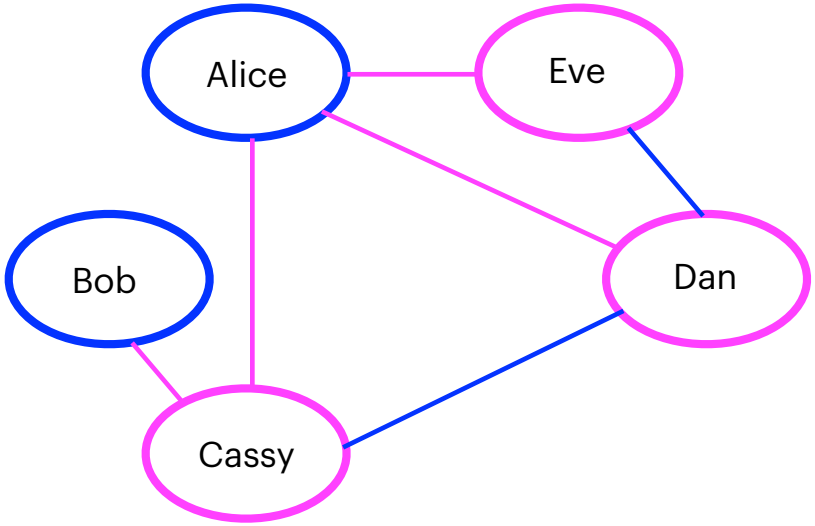


‘+’ : agreement

‘—‘ : disagreement

No link means absence of communications/interaction

A Signed *Opinion* Graph

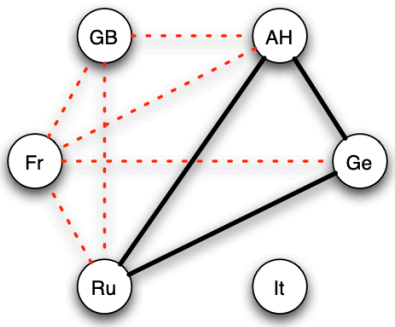


‘+’ : agreement

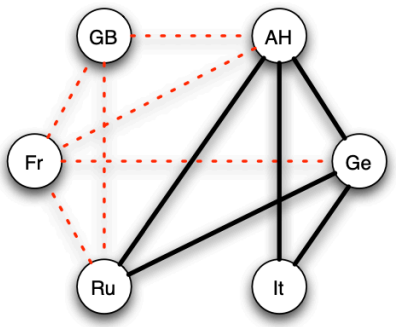
‘—‘ : disagreement

Liberals vs. Conservatives? ☺

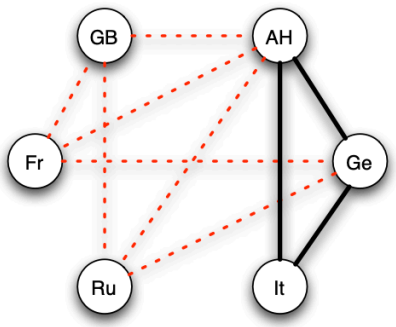
Application



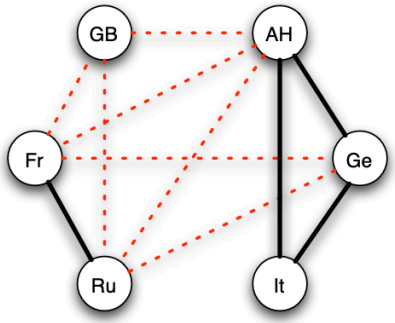
(a) *Three Emperors' League 1872–81*



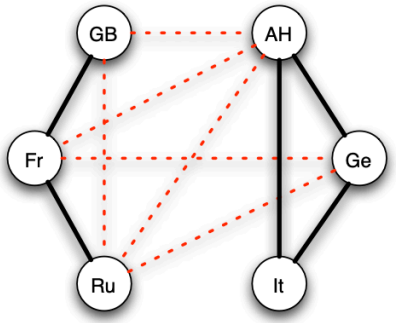
(b) *Triple Alliance 1882*



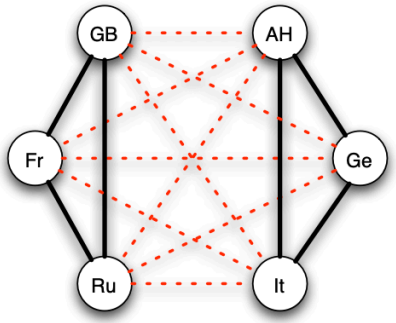
(c) *German-Russian Lapse 1890*



(d) *French-Russian Alliance 1891–94*



(e) *Entente Cordiale 1904*



(f) *British Russian Alliance 1907*

- The nations GB, Fr, Ru, It, Ge, and AH are Great Britain, France, Russia, Italy, Germany, and Austria-Hungary respectively). Solid dark edges indicate friendship while dotted red edges indicate enmity. Note how the network slides into a balanced labeling — and into World War I.
- This figure and example are from Antal, Krapivsky, and Redner [20].

Trust, Distrust and Online Ratings

- A growing source for network data with both positive and negative edges comes from user communities on the Web.
- On Slashdot, users can designate each other as a "friend" or a "foe".
- On Epinions, users can express evaluations of different products, and also express trust or distrust of other users.

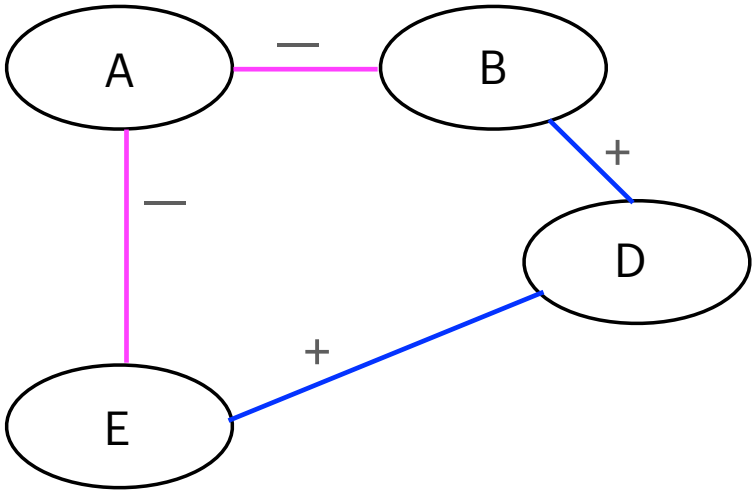
- Epinions is a directed graph.
- When a user A expresses trust or distrust of a user B, we don't necessarily know what B thinks of A, or whether B is even aware of A.

- How do we expect triangles in Epinions to behave?

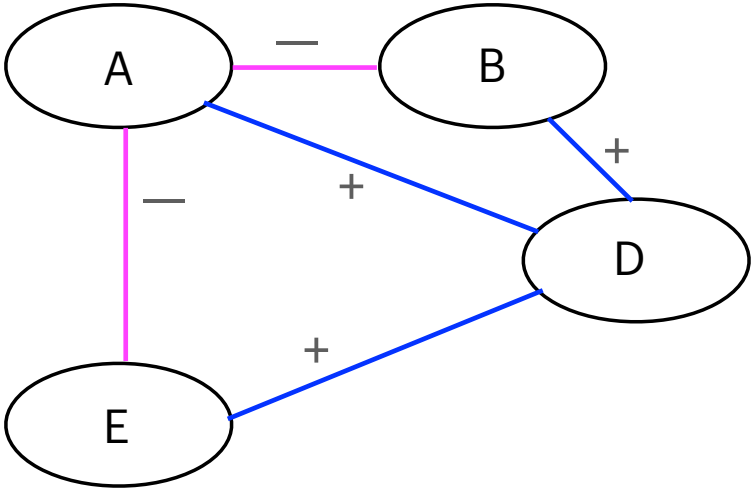
Balance in Signed Graphs

COMPLEX NETWORKS

- A cycle is balanced if it contains an even number of '—' edges.
- G is balanced if each cycle in G is balanced.



Balanced

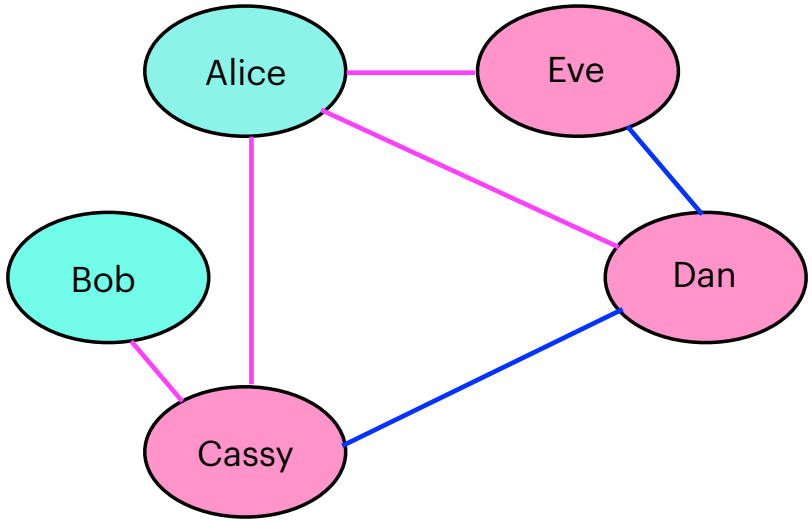


Unbalanced cycles: ABDA & ADEA

Balance in Signed Graphs

COMPLEX NETWORKS

- This is the same as saying we can partition the vertex set V of G into two disjoint subsets:
  - each '+' edge connects two vertices in the same subset,
  - each '—' edge connects two vertices in two different subsets.



$V_1 = \{$ Alice, Bob $\}$

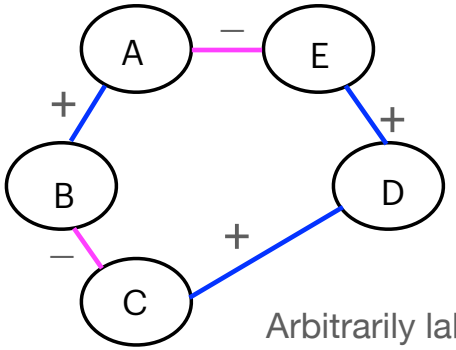
$V_2 = \{$ Cassy, Dan, Eve $\}$

balanced graph

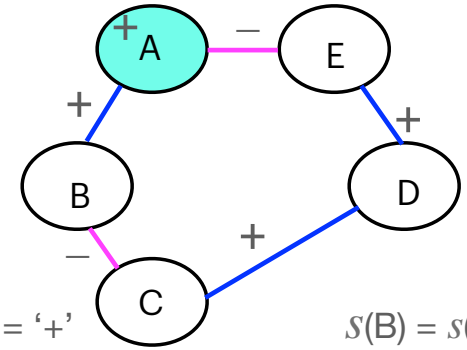
Assigning Node Labels: *Balanced Graph*

COMPLEX NETWORKS

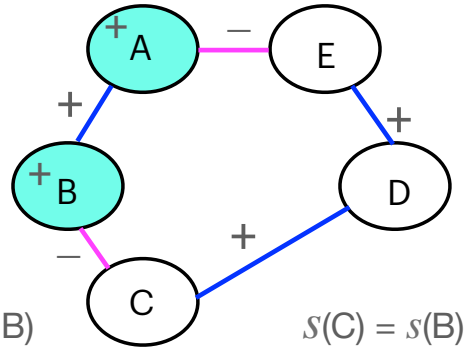
✳ Based on edge labels $s(x, y)$, we can assign node labels $s(x)$.



Arbitrarily label s(A) = '+'

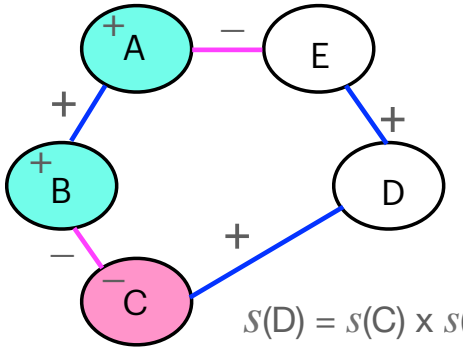


s(B) = s(A) x s(A,B)

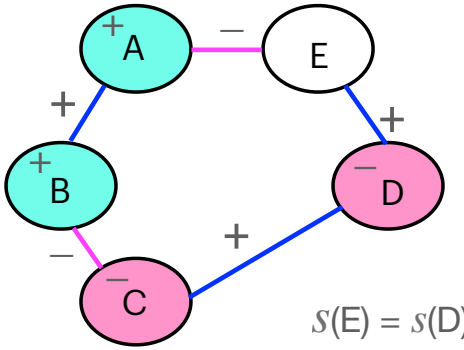


s(C) = s(B) x s(B,C)

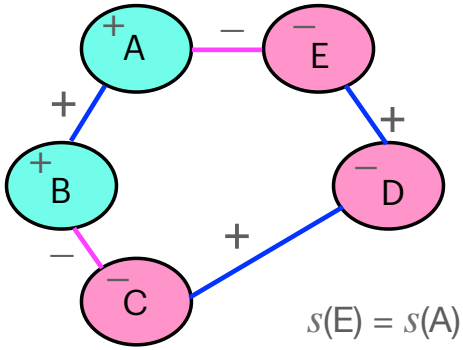
Order does not matter!



s(D) = s(C) x s(C,D)



s(E) = s(D) x s(D,E)



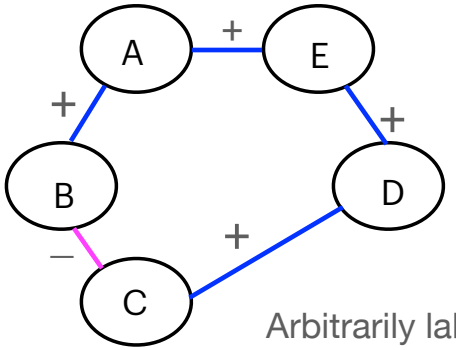
s(E) = s(A) x s(A,E)

Background

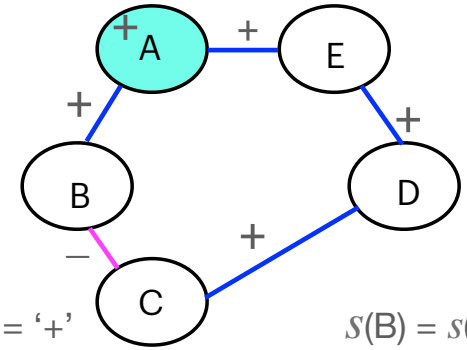
Assigning Node Labels: *Unbalanced Graph*

COMPLEX NETWORKS

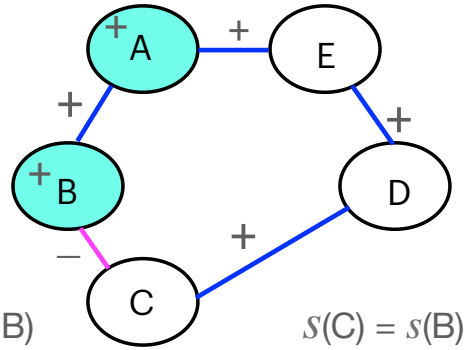
✳ *It is not possible to assign node labels $s(x)$ consistently in an unbalanced graph.*



Arbitrarily label s(A) = '+'

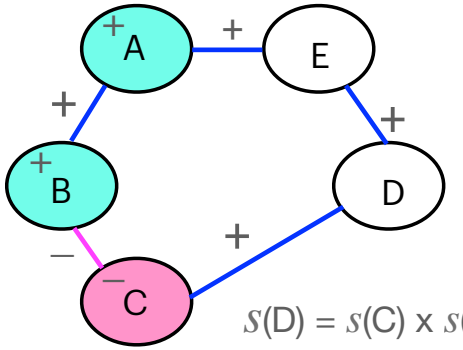


s(B) = s(A) x s(A,B)

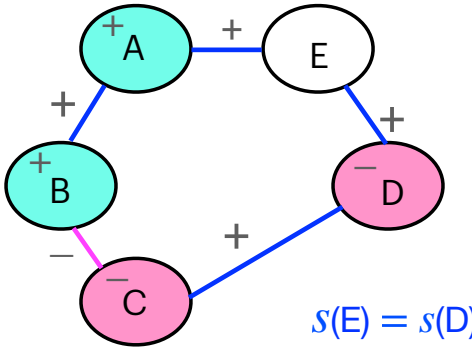


s(C) = s(B) x s(B,C)

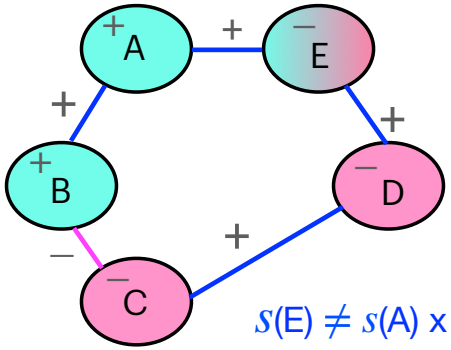
Conflict will appear somewhere!



s(D) = s(C) x s(C,D)

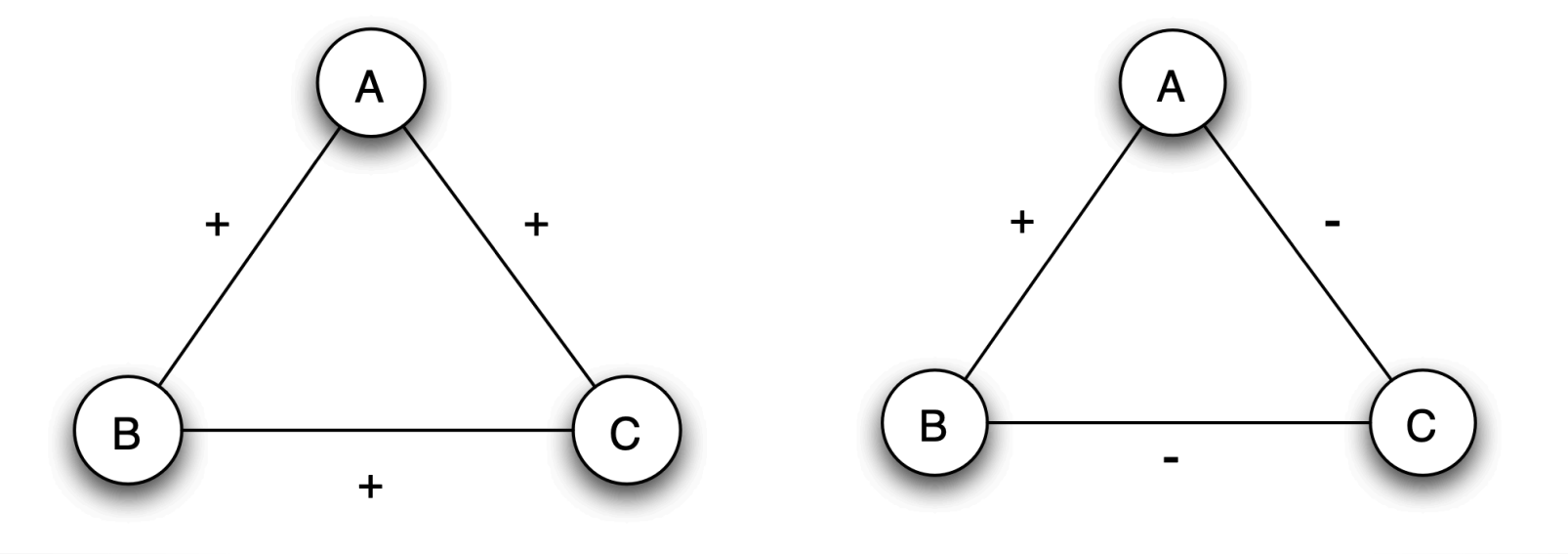


s(E) = s(D) x s(D,E)

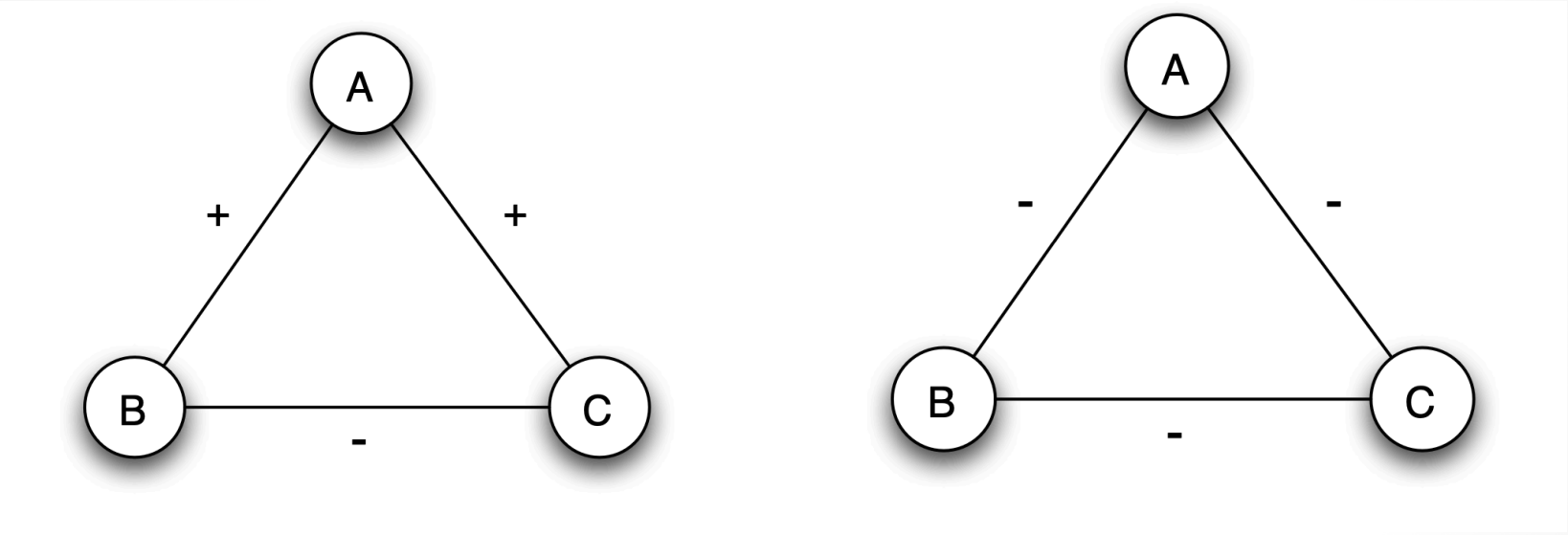


s(E) ≠ s(A) x s(A,E)

Structural Balance



Structural Balance



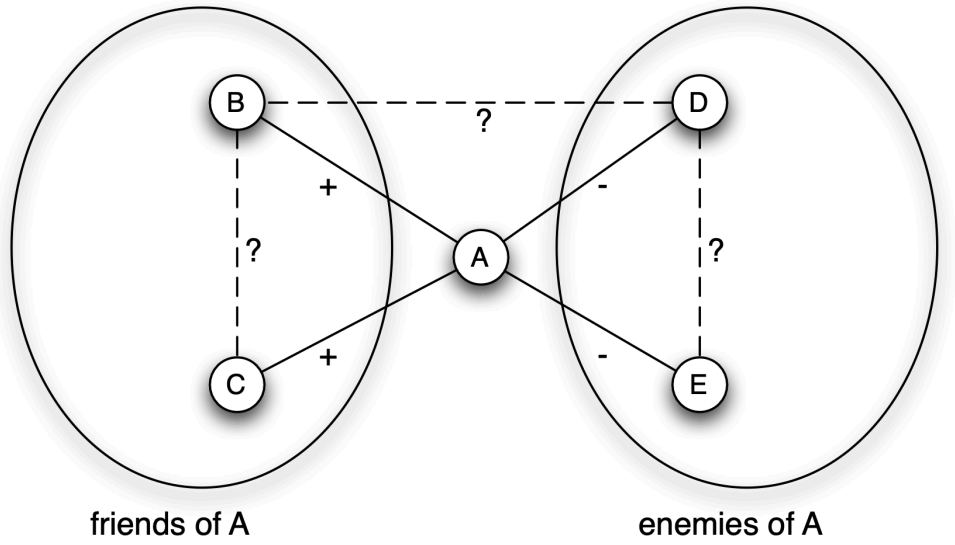
Balance Theorem

- So this describes two basic ways to achieve structural balance:
  - either everyone likes each other; or
  - the world consists of two groups of mutual friends with complete antagonism between the groups.

- The surprising fact is the following:
  these are the only ways to have a balanced network.

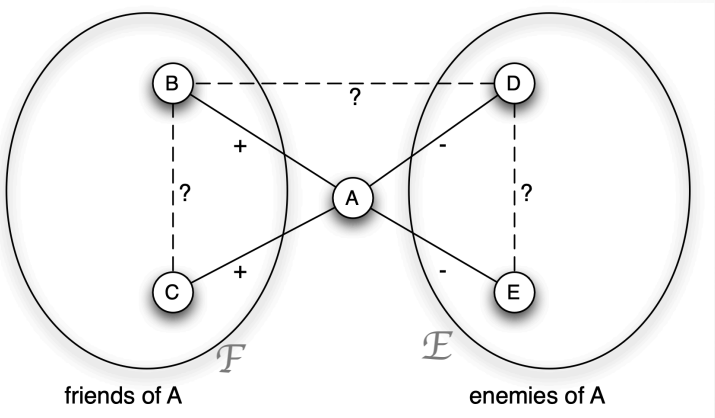
Balance Theorem

- *Balance Theorem: If a labeled complete graph is balanced, then either all pairs of nodes are friends, or else the nodes can be divided into two groups, X and Y , such that every pair of nodes in X like each other, every pair of nodes in Y like each other, and everyone in X is the enemy of everyone in Y.*



Proof of the Balance Theorem

- (Given) Suppose we have a labeled complete graph, and all we know is that it's balanced.
- (To prove that) Everyone in each group are friends with each other and are enemies with the other group.
- If it has no negative edges, we are done.
- Else, there is at least one negative edge.
- Consider some node A. Since this is a complete graph, every node is either a friend or an enemy of A.
- Let the friends of A be in set $\mathcal{F}$, and enemies in set $\mathcal{E}$.
- Since each triangle is balanced,
  B-C must be +
  D-E must be +
  B-D must be -
- □



Structural Balance

- This illustrates a nice connection between local and global network properties.
- A recurring issue in the analysis of networked systems is the way in which local effects — phenomena involving only a few nodes at a time — can have global consequences that are observable at the level of the network as a whole.
- The notion of Structural balance offers a way to capture one such relationship in a very clean way, and by purely mathematical analysis.

Balancing Unbalanced Graphs

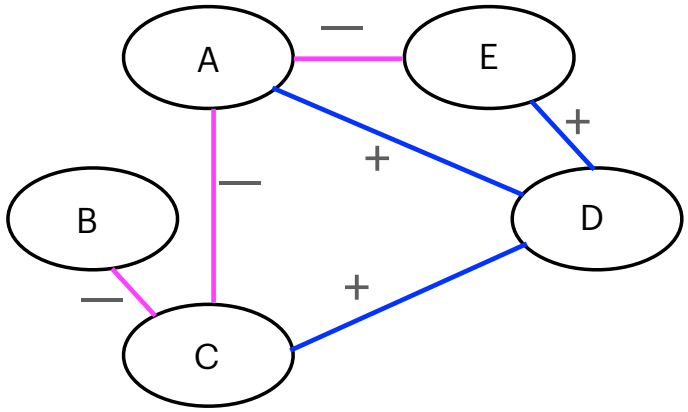
"A More Powerful Heuristic for Balancing an Unbalanced Graph" Intl. Conf. on Complex Networks and Their Applications, 2022.

"A Correction to The Heuristic Algorithm MinimalFlipSet to Balance Unbalanced Graphs", Intl. Conf. on Complex Networks and Their Applications, 2023.

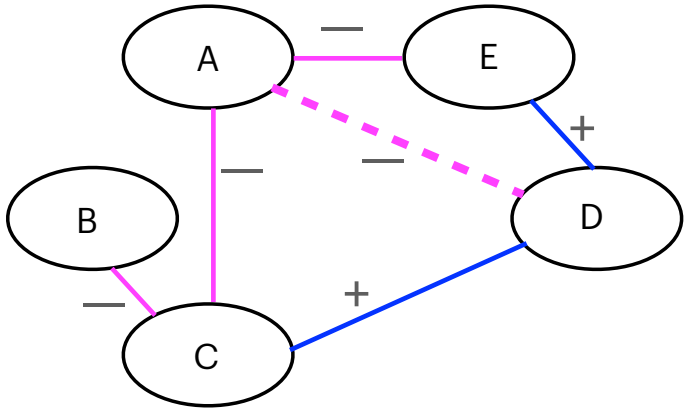
Our Problem: *Balancing* Unbalanced Graphs

COMPLEX NETWORKS

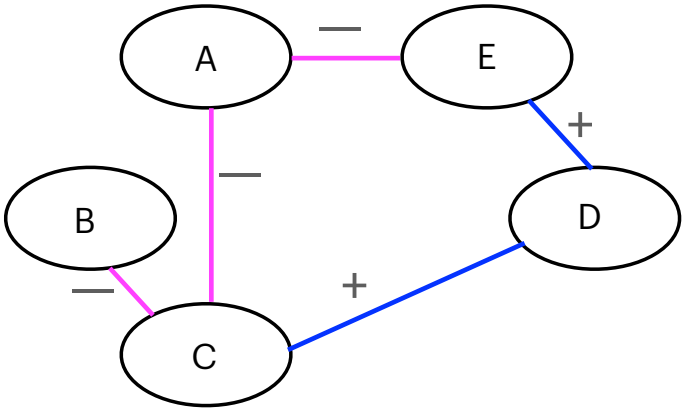
- Two ways to balance an unbalanced graph:
  - flipping the signs of a subset of edges
  - deleting a subset of edges



unbalanced

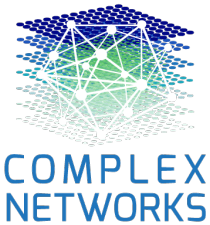


balanced by flipping (A,D)



balanced by deleting (A,D)

Balancing Unbalanced Graphs:
What we already know



✳ $E_{optFlip}$: The optimal (minimum size) flipping edge-sets

✳ E_{optDel} : The optimal (minimum size) deletion edge-sets

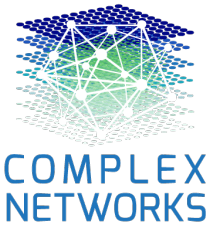
$$|E_{optFlip}| = |E_{optDel}|$$

✳ Finding an E_{optDel} $\simeq$ Finding a maximum size bipartite subgraph of G^-
(the subgraph of negative links in G).

✳ Finding maximum bipartite subgraph of a general G (and hence of G^-) is *NP-complete*.

✳ Finding an $E_{optFlip}$ for a general signed graph G is also *NP-complete*.

Balancing Unbalanced Graphs:
Our Goal

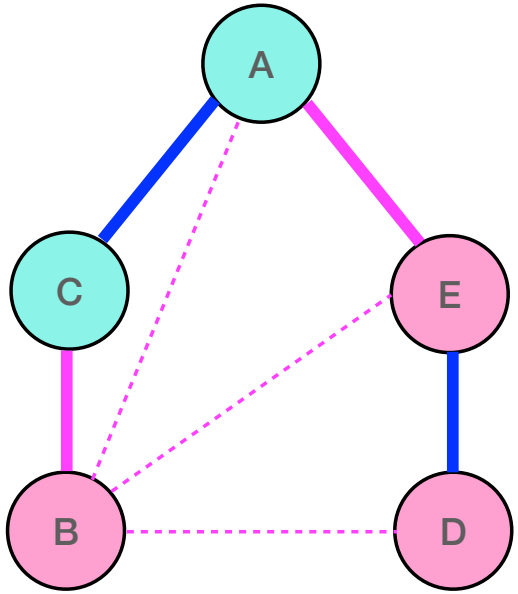


- Finding an $E_{optFlip}$ for a general signed graph G is *NP-complete*.
- Find $E_{malFlip}$ instead.

How to check if a graph is balanced?

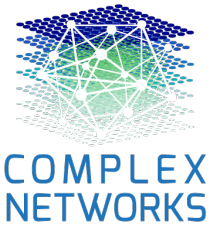
COMPLEX NETWORKS

- Find a fixed rooted spanning tree T of G.
- Assign node labels based on T.
- For each non-tree edge (x, y), we must have $s(x)s(y) = s(x, y)$;
  Any violation means G is unbalanced
  - ACBA: s(A,B) = s(A) x s(B) ✓
  - ACBEA: s(B,E) = s(B) x s(E) ✗
  - ACBDEA: s(B,D) = s(B) x s(D) ✗



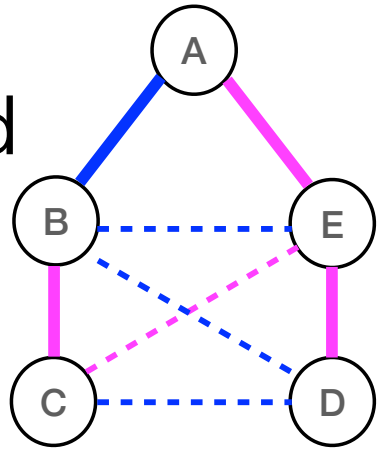
A single violation tells us that G is unbalanced!

Balancing Unbalanced Graphs:
Our Goal



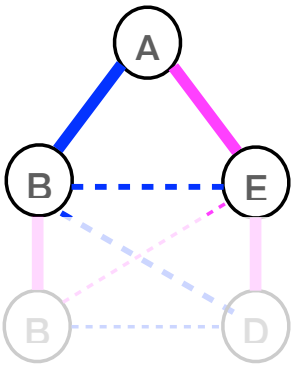
✳ Find $E_{malFlip}$ a minimal set of edges to flip in order to balance G.

AL21's Idea*: Flip non-tree edges as needed

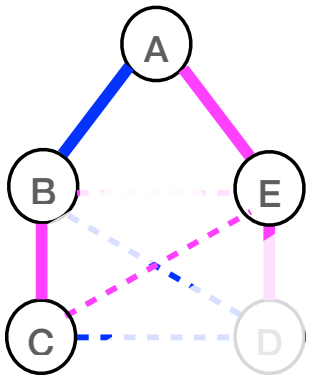


COMPLEX NETWORKS

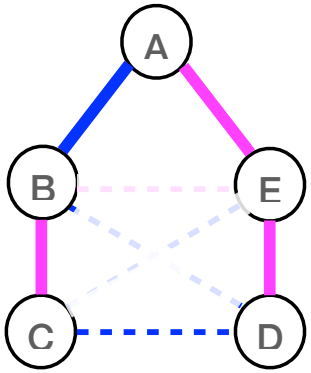
- For each fundamental cycle due to a non-tree edge (x, y),
  - If the cycle is unbalanced, flip $s(x, y)$



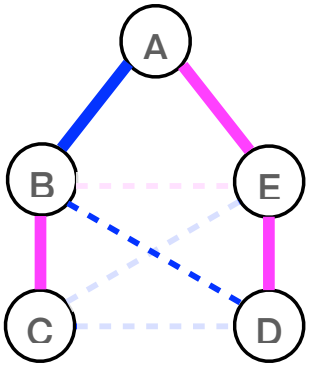
flip (B,E)



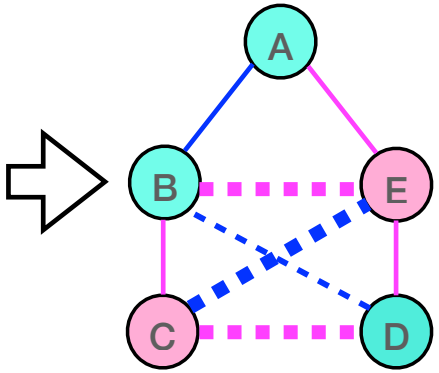
flip (C,E)



flip (C,D)



do not flip (B,D)



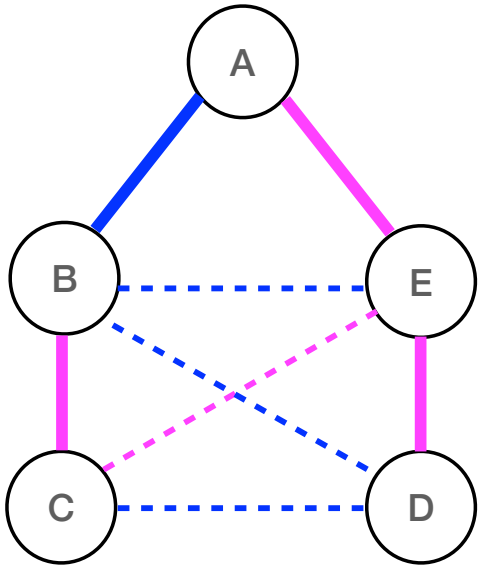
Balanced in 3 flips!

* G. Alabandi et al., Discovering and balancing fundamental cycles in large signed graphs, Proc. Int'l Conf for HPCNSA, 2021, pp. 1–17.

A more efficient version of AL21's algorithm

COMPLEX NETWORKS

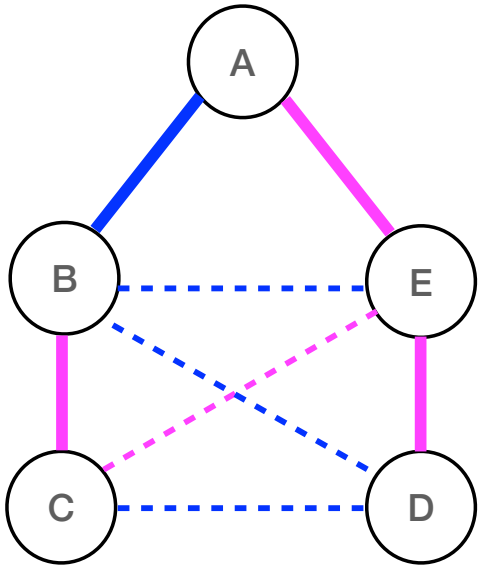
- Traverse the spanning tree T in a depth-first manner and label the nodes, starting with the label '+' for root.
- For each node x (chosen in any order), process all edges $(x, y) \notin T$ from x (i.e., $y \neq$ *parent(x)* and $x \neq$ *parent(y)* as follows:
  - *if* $s(x, y) \neq s(x)s(y)$, *then* include (x, y) in the flip-set.
- AL21's time complexity is $O(|V||E|)$
- Our version run in $O(|E|)$, giving a speedup of $O(|V|)$ over AL21.



Our Idea: Allow Flipping tree-edges too

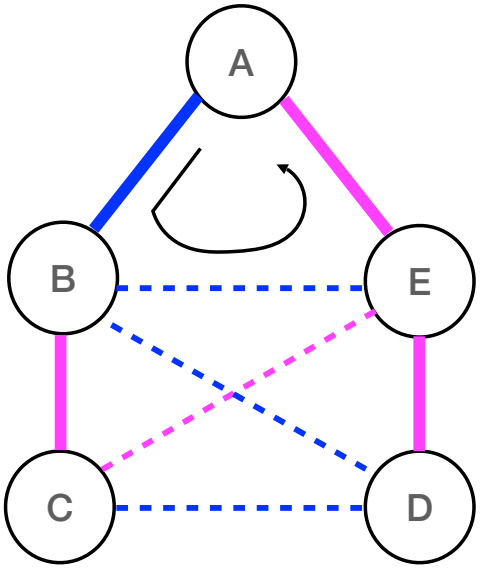
COMPLEX NETWORKS

- Why?
  - Reduce the number of flips



Selecting a tree-edge (u, v) for flipping

- A tree edge (u, v) "covers" all non-tree edges (x, y) whose fundamental cycle contains (u, v).

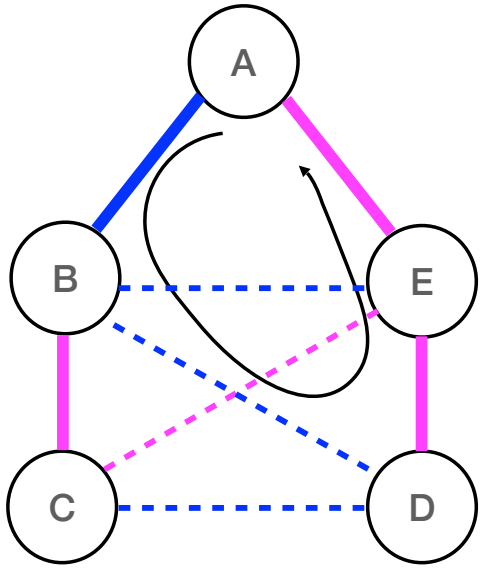


(A, B) covers { **(B, E)**, (B, D), (C, D), (C,E) }

(B, E) is unbalanced: ABEA

Selecting a tree-edge (u, v) for flipping

- A tree edge (u, v) "covers" all non-tree edges (x, y) whose fundamental cycle contains (u, v).

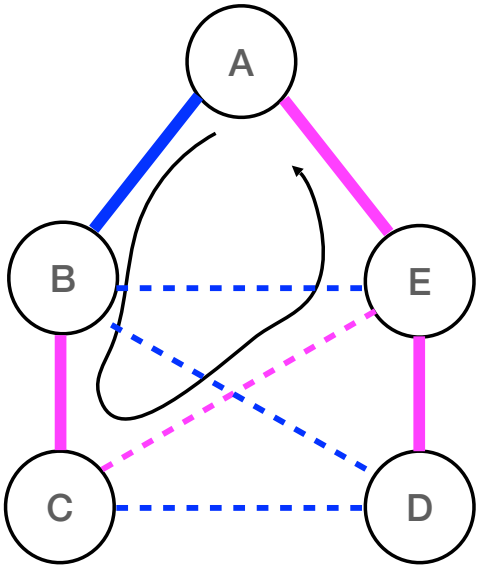


(A, B) covers { (B, E), **(B, D)**, (C, E), (C, D) }

(B, D) is balanced: ABDEA

Selecting a tree-edge (u, v) for flipping

- A tree edge (u, v) "covers" all non-tree edges (x, y) whose fundamental cycle contains (u, v).



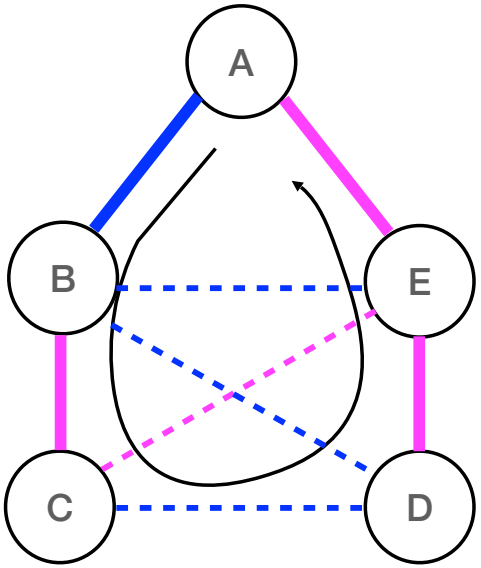
(A, B) covers { (B, E), (B, D), **(C, E)**, (C, D) }

(C, E) is unbalanced: ABCEA

Selecting a tree-edge (u, v) for flipping

COMPLEX NETWORKS

- A tree edge (u, v) "covers" all non-tree edges (x, y) whose fundamental cycle contains (u, v).

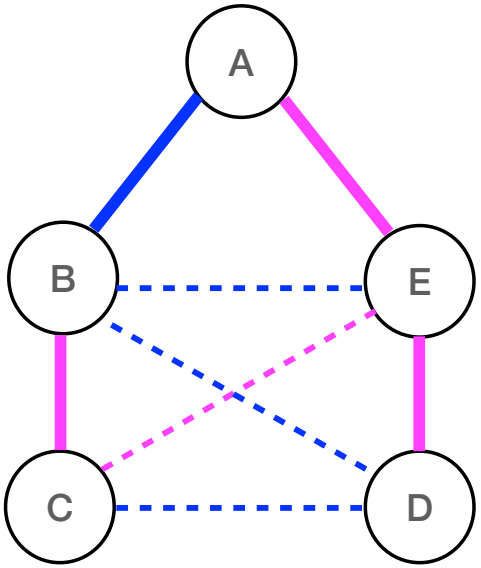


(A, B) covers { (B, E), (B, D), (C, E), **(C, D)** }

(C, D) is unbalanced: ABCDEA

Selecting a tree-edge (u, v) for flipping

- A tree edge (u, v) "covers" all non-tree edges (x, y) whose fundamental cycle contains (u, v).



(A, B) covers { (B, E), (B, D), (C, D), (C,E) }

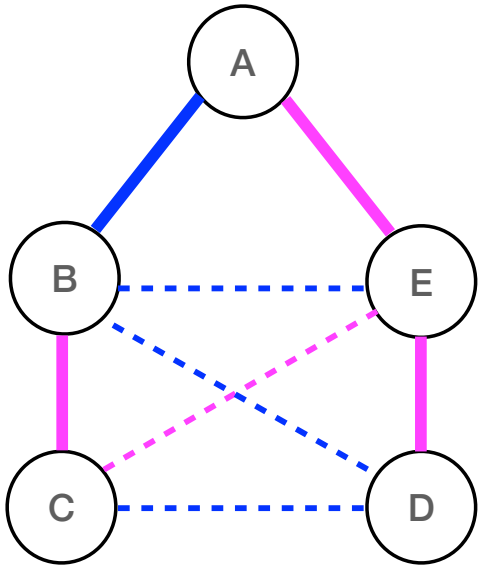
(B, E) is unbalanced: ABEA
(B, D) is balanced: ABDEA
(C, E) is unbalanced: ABCEA
(C, D) is unbalanced: ABCDEA

Selecting a tree-edge (u, v) for flipping

- A tree edge (u, v) "covers" all non-tree edges (x, y) whose fundamental cycle contains (u, v).

(A, B) covers { (B, E), (B, D), (C, D), (C,E) }

(B, E) is unbalanced: ABEA
(B, D) is balanced: ABDEA
(C, E) is unbalanced: ABCEA
(C, D) is unbalanced: ABCDEA



$$\xi(A,B) = \{(B,E),(B,D),(C,D),(C,E)\}$$

$$\xi_b(A,B) = \{(B,D)\}$$

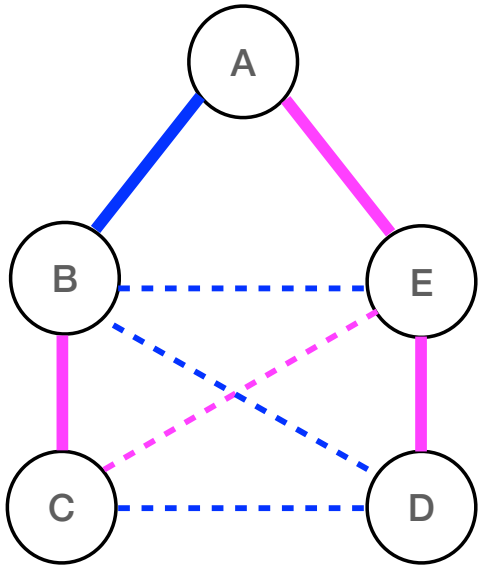
$$\xi_u(A,B) = \{(B,E),(C,D),(C,E)\}$$

✳ Is flipping (A, B) a good idea?
- Flipping (A, B) will balance 3 unbalanced fundamental cycles, and imbalances 1 cycle.

Selecting a tree-edge (u, v) for flipping

(A, B) covers { (B, E), (B, D), (C, D), (C,E) }

(B, E) is unbalanced: ABEA
(B, D) is balanced: ABDEA
(C, E) is unbalanced: ABCEA
(C, D) is unbalanced: ABCDEA



$\xi_c(A, B) = \{(B, E), (B, D), (C, D), (C, E)\}$

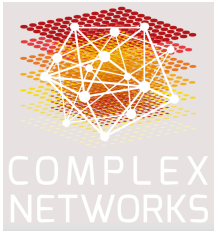
$\xi_b(A, B) = \{(B, D)\}$

$\xi_u(A, B) = \{(B, E), (C, D), (C, E)\}$

✳ Condition for flipping $(u, v) \in T$:

- $|\xi_b(u, v)| + 1 \leq |\xi_u(u, v)|$
- $gain(u, v) = |\xi_u(u, v)| - |\xi_b(u, v)| \geq 1$

Algorithm NewFlipSet



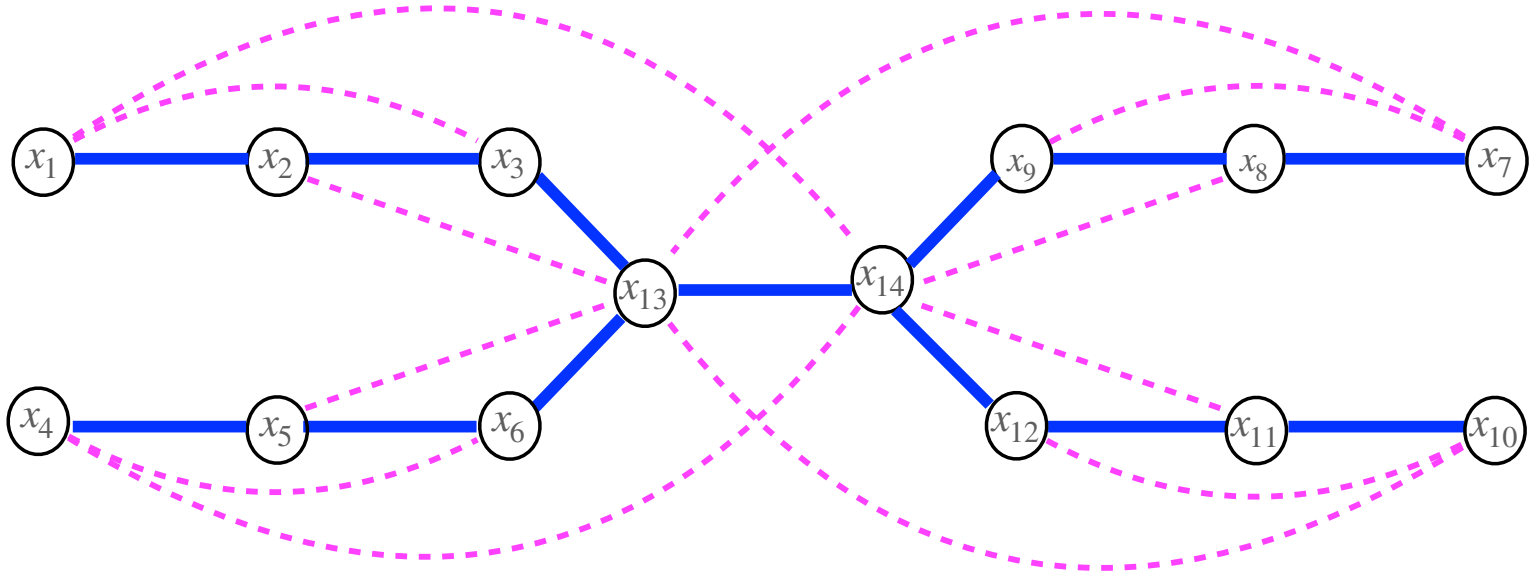
✳ Input: A connected signed graph $G = (V, E)$ and a spanning tree T of G.

✳ Output: A flipping edge-set $E_{smallFlip}$ consisting of edges possibly from both T and $(E - T)$ for balancing G.

1. Initialise $E_{smallFlip} = \emptyset$.
2. For each edge $(u, v) \in T$, calculate $gain(u, v)$.
3. Repeat steps (a) — (c):
   (a) Let $M = max\{gain(u, v) : (u, v) \in T\}$, and let (u', v') an edge in T with $gain(u', v') = M$.
   (b) If $(M \geq 1)$,
      (i) Modify G by flipping the label of the edges (u', v') and all the (non-tree) edges covered by it.
      Add (u', v') to $E_{smallFlip}$ if it is not already in it, else remove it from $E_{smallFlip}$.
      (ii) Let $T(u', v')$ be the tree-edges that cover the non-tree edges covered by (u', v').
      Recompute $\xi_u(e)$, $\xi_b(e)$ and $gain(e)$ for all edges $e \in T(u', v')$.
   while $(M \geq 1)$
4. For each remaining unbalanced $(x, y) \notin T$, add (x, y) to $E_{smallFlip}$.

Illustration of Algorithm NewFlipSet

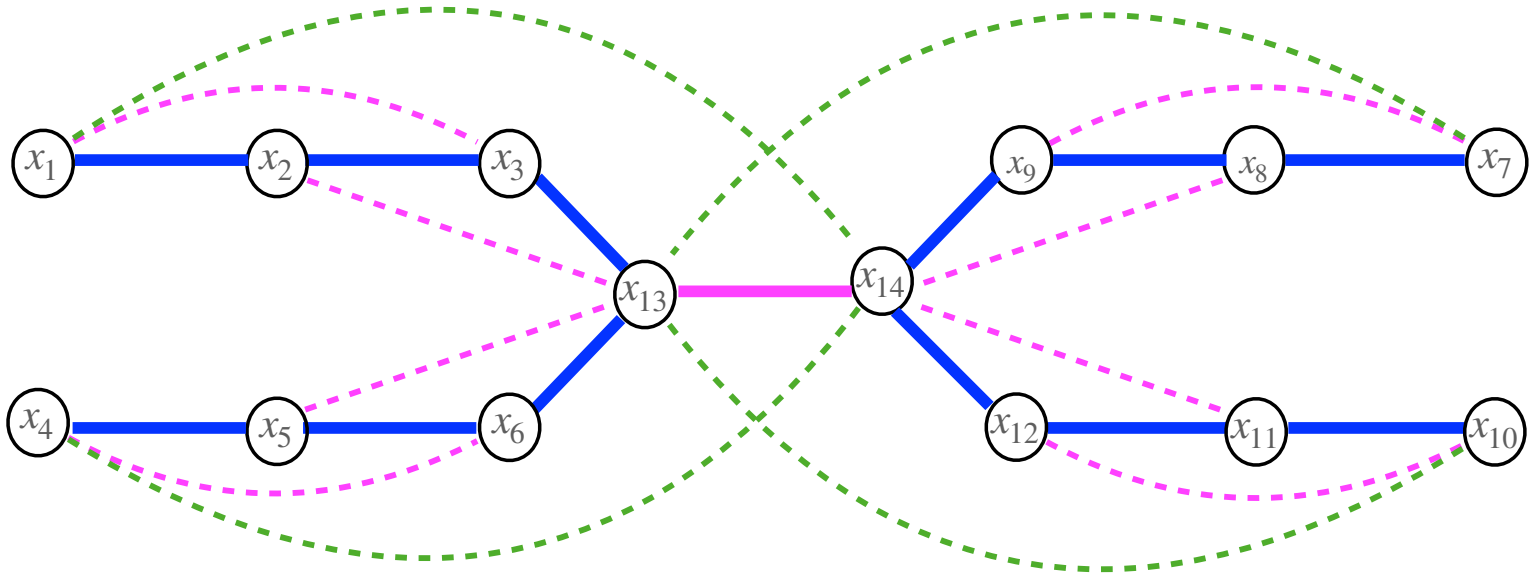
✳ Algorithm NewFlipSet can flip a tree-edge *multiple* times (as in the old algorithm MinimalFlipSet).



Each pink (non-tree) edge gives an unbalanced fundamental cycle.

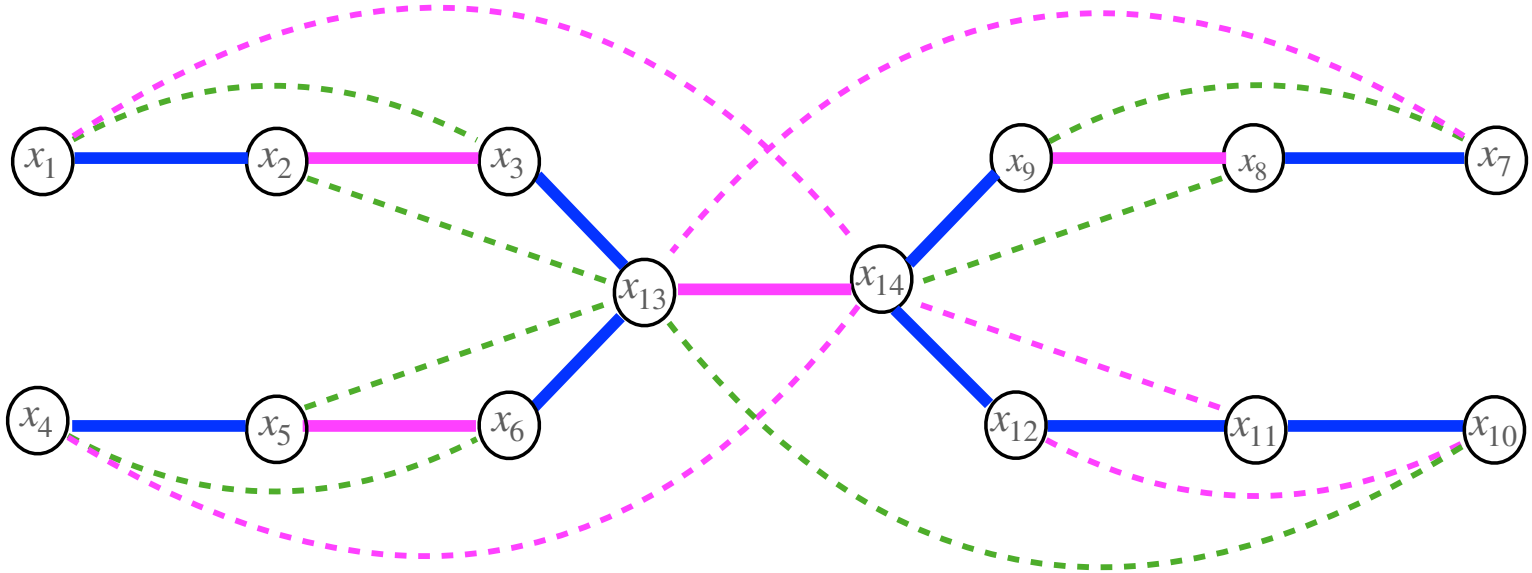
The best tree-edge to flip is (x_{13}, x_{14}).

Illustration of Algorithm NewFlipSet



The graph after flipping (x_{13}, x_{14}), with maximum gain M = 4.
Each green (non-tree) edge gives a balanced fundamental cycle.

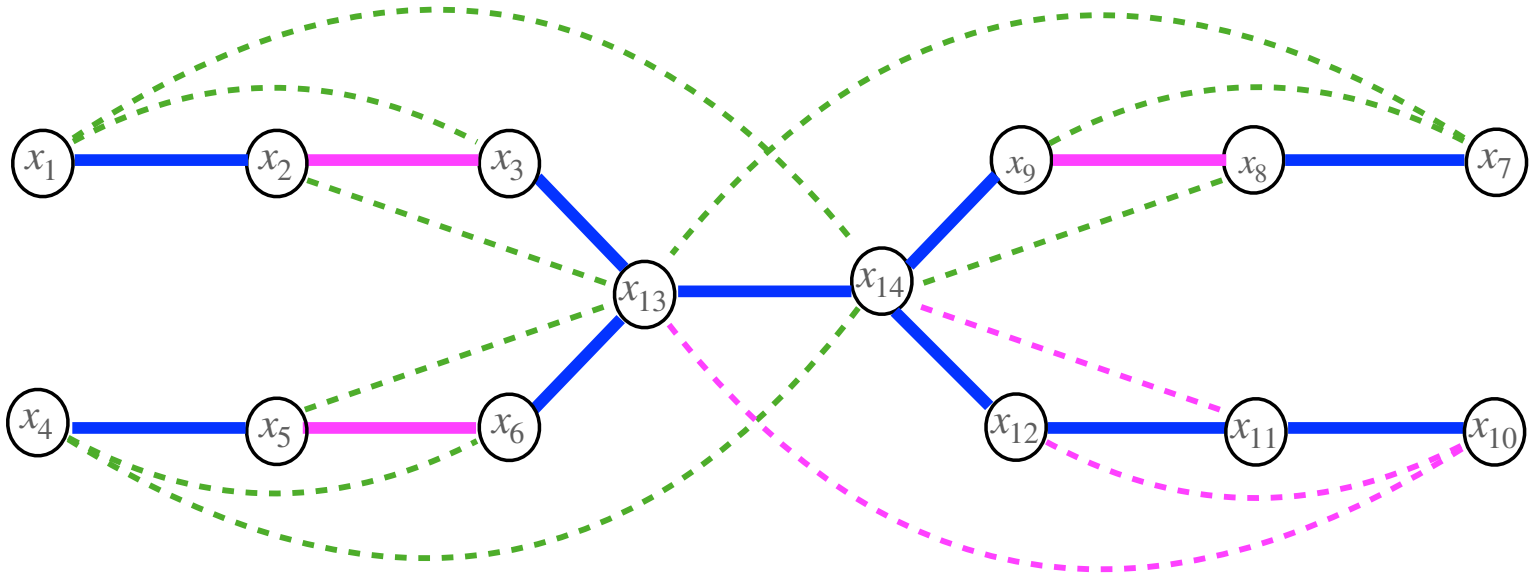
Illustration of Algorithm NewFlipSet



The graph after flipping the next non-tree edges $(x_2, x_3), (x_5, x_6), (x_8, x_9)$ with maximum gain M = 1 each time.

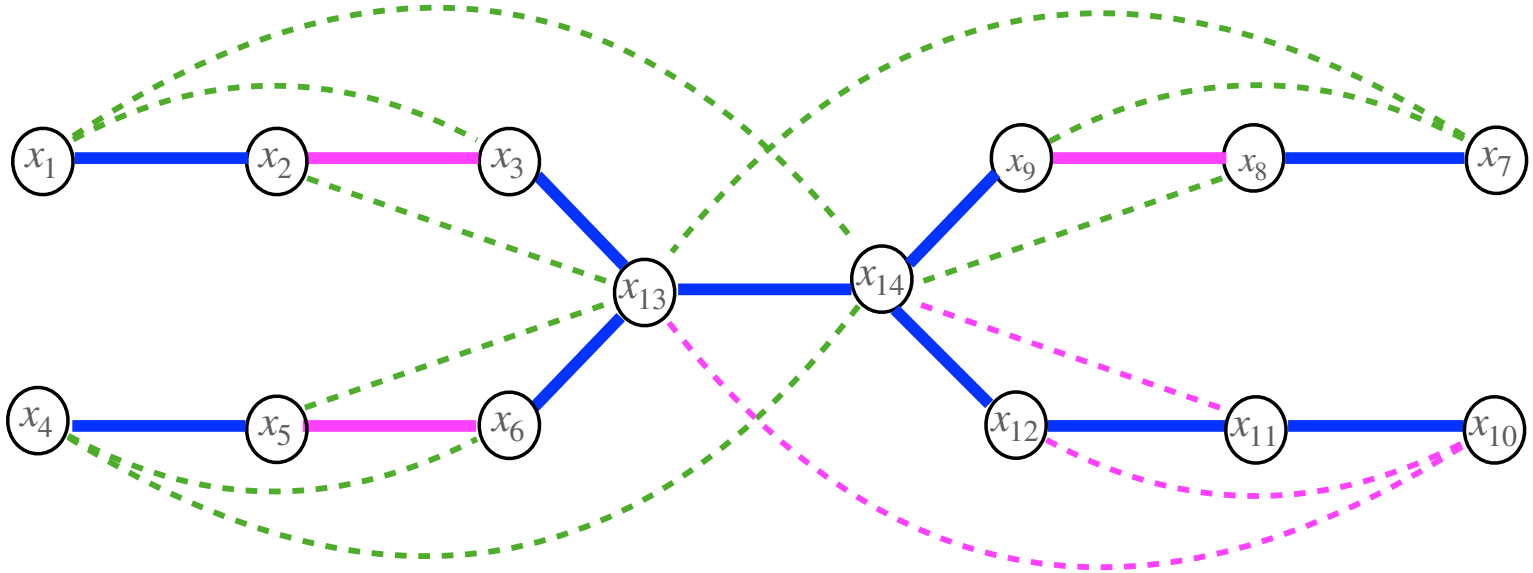
The next best tree-edge to flip is (x_{13}, x_{14}) *again,* with gain M = 2.

Illustration of Algorithm NewFlipSet



The result of flipping tree-edge (x_{13}, x_{14}).

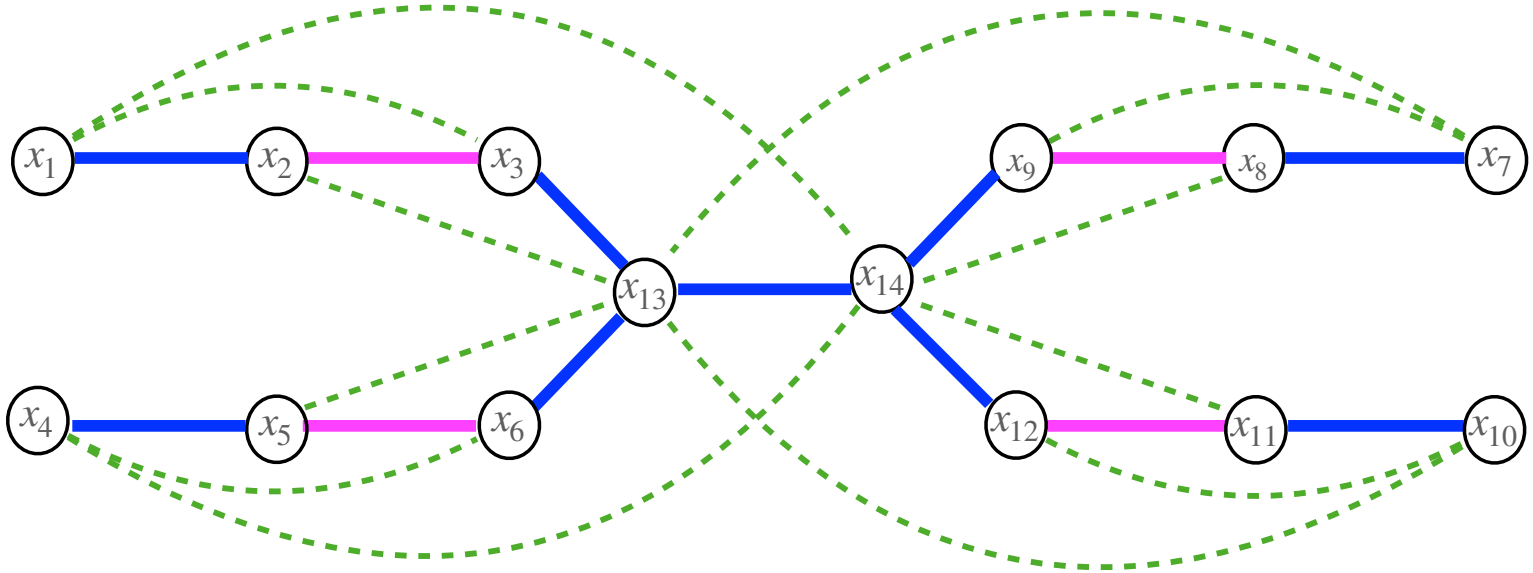
Illustration of Algorithm NewFlipSet



Since (x_{13}, x_{14}) is flipped twice, it is excluded from $E_{smallFlip}$.

The best tree-edge to flip is now (x_{11}, x_{12}), with gain M = 3.

Illustration of Algorithm NewFlipSet



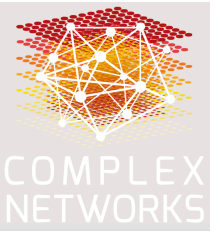
The balanced graph after flipping (x_{11}, x_{12}).

$$E_{smallFlip} = E_{optFlip} = \{(x_2, x_3), (x_5, x_6), (x_8, x_9), (x_{11}, x_{12})\}$$

This original graph (page 26) has at least 4 edge-disjoint unbalanced cycles like $x_1x_2x_3x_1$, $x_4x_5x_6x_4$, and similarly for x_7 and x_{10}.

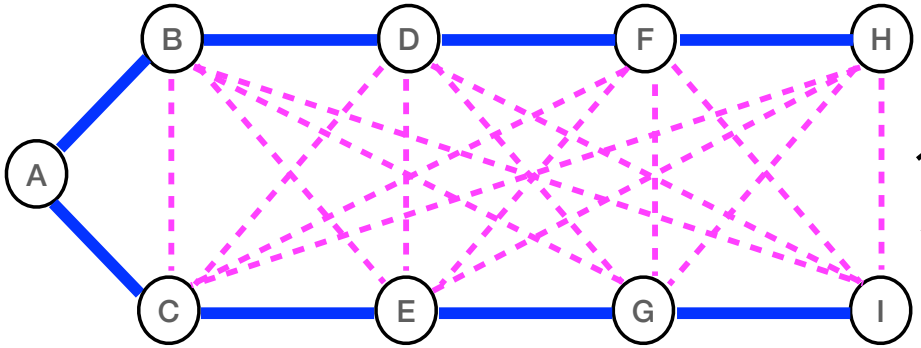
Thus, our $E_{smallFlip}$ of size 4 $\leq |E_{optFlip}|$ is an $E_{optFlip}$.

Algorithm NewFlipSet

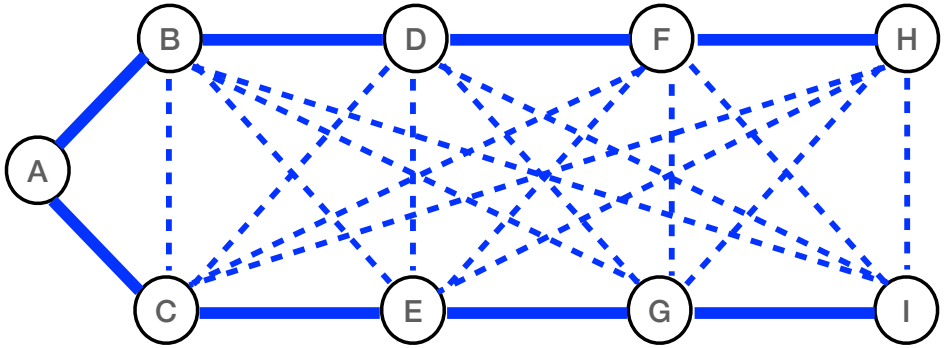


- Algorithm NewFlipSet excludes a tree-edge that is flipped an even number of times, unlike Algorithm MinimalFlipSet.
- It can give a smaller flipping edge-set than Algorithm MinimalFlipSet.
- Proof of termination provided for Algorithm NewFlipSet.

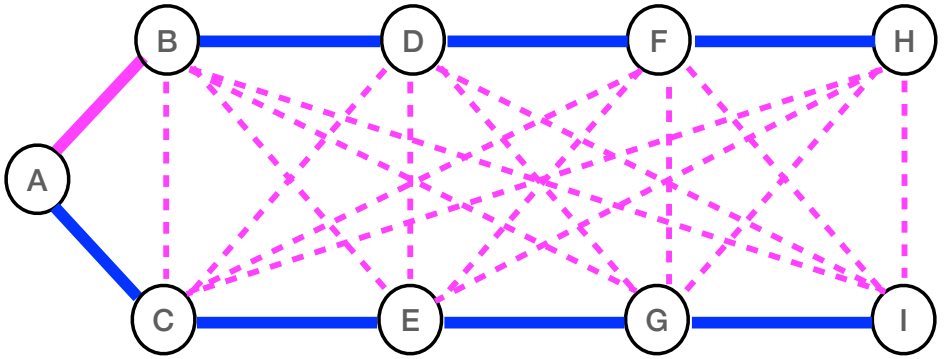
Our method vs. AL21



16 flips
AL '21



1 flip
our
method



We could get as much as $O(n^2)$ improvement in the size of the solution over AL21!

Ahmedabad University

Thank you!

Amit A. Nanavati
Ahmedabad University