

Linear Systems and k -Dimensional Weisfeiler-Lehman Stabilization

Chris Augeri^{1*}, Barry Mullins¹, Leemon Baird²,
Rusty Baldwin¹, & Dursun Bulutoglu¹

¹Air Force Institute of Technology, Dayton, OH

²United States Air Force Academy, Colorado Springs, CO

Overview

- **UAV Swarms**
- **Graph Canonization**
- **Graph Isomorphism**
- **k -D Weisfeiler-Lehman Stabilization via**
 - discrete k -tuples (modern view)
 - matrix multiplication (Weisfeiler-Lehman)
 - linear systems (our work)

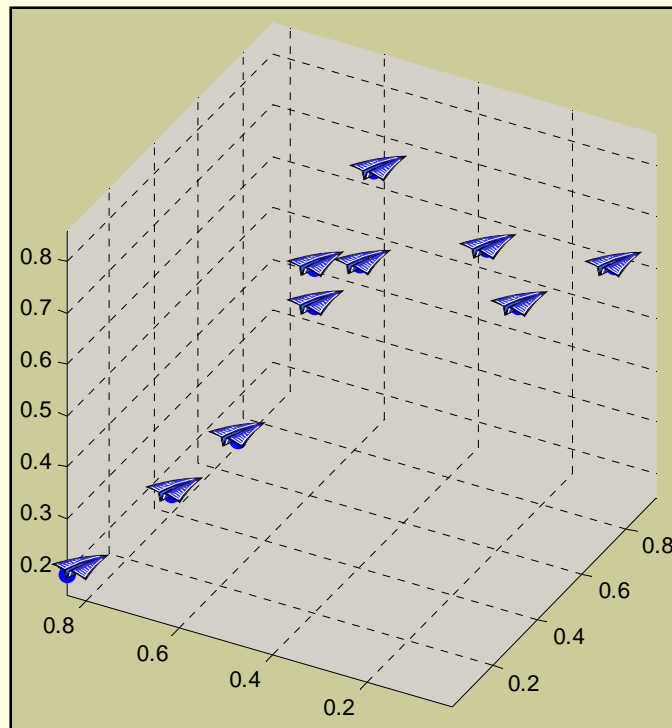
We Aren't Planning to Discuss...

If the graph isomorphism problem (GIP) is

- $\in P, BPP, NP\text{-Complete}, \dots?$
- captured by a formal language, X , e.g., first-order predicate logic?
- solved using polynomial time by a quantum algorithm on a quantum computer?

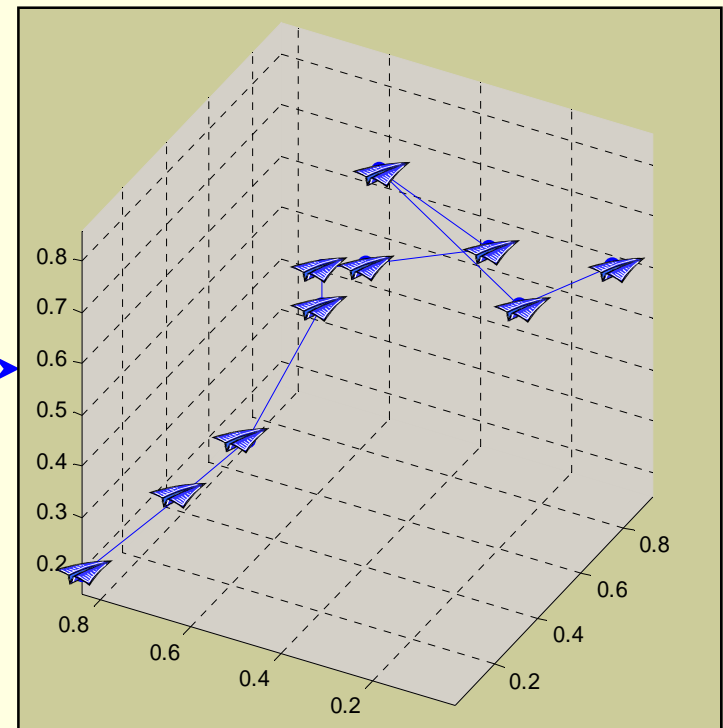
Unmanned Aerial Vehicle (UAV) Swarms

- **Dimensions:** coordinates, time, values, ...
- **Applications:** queries, logistics, security, ...
- **Orderings:** space-filling curves, k -D trees, MDS, ...



UAV Swarm

compute
—→
ordering



Potential Linearization 4

Graphs & Linear Algebra

0 or **Z** (**z**) all-zero matrix (vector)

1 or **J** (**j**) all-ones matrix (vector)

A adjacency matrix of G

I identity matrix

P permutation matrix

X information matrix

$$\mathbf{P} \cdot \mathbf{A} \cdot \mathbf{P}^T \leftrightarrow \mathbf{P} \cdot \mathbf{X} \cdot \mathbf{P}^T$$

$$\mathbf{A}_1 \cong \mathbf{A}_2 \leftrightarrow \mathbf{X}_1 \cong \mathbf{X}_2$$

D (**d**) degree matrix (vector)

$$\mathbf{D}^{n,n} = \mathbf{0}, \mathbf{D}_{i,i} = \mathbf{d}_i = \deg(v_i)$$



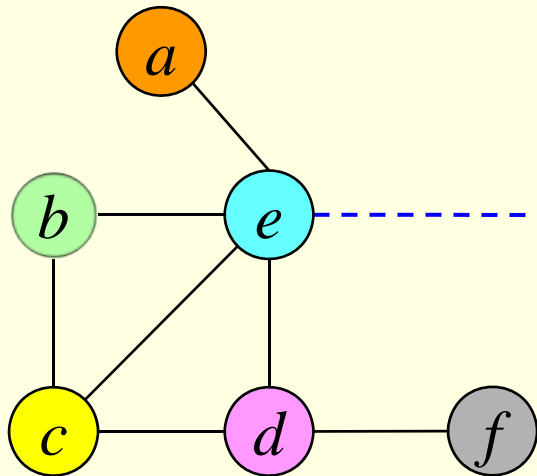
	u	v	w
u	0	1	0
v	1	0	1
w	0	1	0

Perron-Frobenius Theorem

- Given a *positive* matrix, \mathbf{A}
 - $\mathbf{A}_{i,j} > 0, \forall i, j$
 - \mathbf{A} has a positive real eigenvalue, λ_n
 - \mathbf{x}_n is *the* positive eigenvector associated with λ_n
- The PageRank algorithm
 - perturbs \mathbf{A} to obtain $\mathbf{A}'_{i,j} > 0$,
 - where $\mathbf{A}' = \alpha \cdot \mathbf{D}^{-1} \cdot \mathbf{A} + (1 - \alpha) / n$,
 - hence, \mathbf{A}' is row-stochastic
 - therefore, $\lambda_n = 1$ & \mathbf{x}_n is \mathbf{A}' 's stationary distribution

PageRank & Canonical Isomorphs

- Heart of Google's web page rankings
- Ranks pages by leading eigenvector [PBM+99]
- Yields canonical isomorph on most random graphs



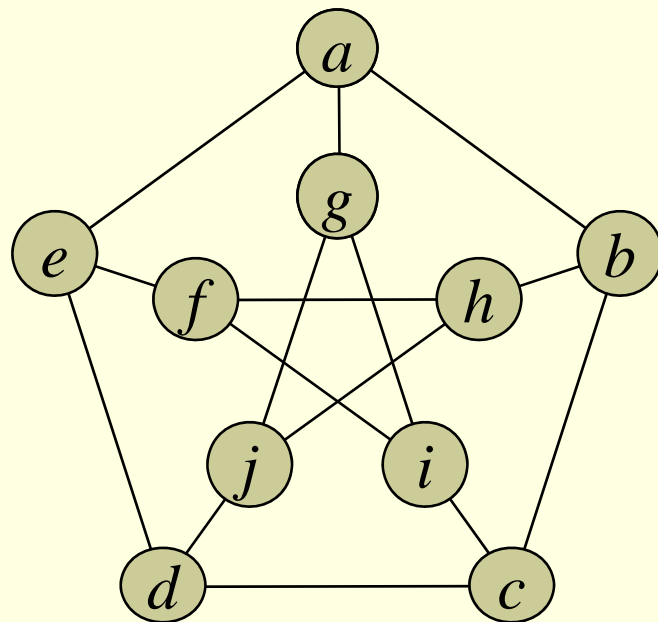
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
<i>a</i>	0	0	0	0	1	0
<i>b</i>	0	0	1	0	1	0
<i>c</i>	0	1	0	1	1	0
<i>d</i>	0	0	1	0	1	1
<i>e</i>	1	1	1	1	0	0
<i>f</i>	0	0	0	1	0	0

λ	1.000
<i>a</i>	0.188
<i>b</i>	0.318
<i>c</i>	0.460
<i>d</i>	0.482
<i>e</i>	0.618
<i>f</i>	0.193

PageRank & Non-Canonical Isomorphs

- **Extreme** cases, *all* eigenvector entries equal.
- **Thus**, the ordering is unlikely to be canonical
- **Can** we find a more useful information matrix, \mathbf{X}

Petersen Graph



λ	1.00
a	0.32
b	0.32
c	0.32
d	0.32
e	0.32
f	0.32
g	0.32
h	0.32
i	0.32
j	0.32

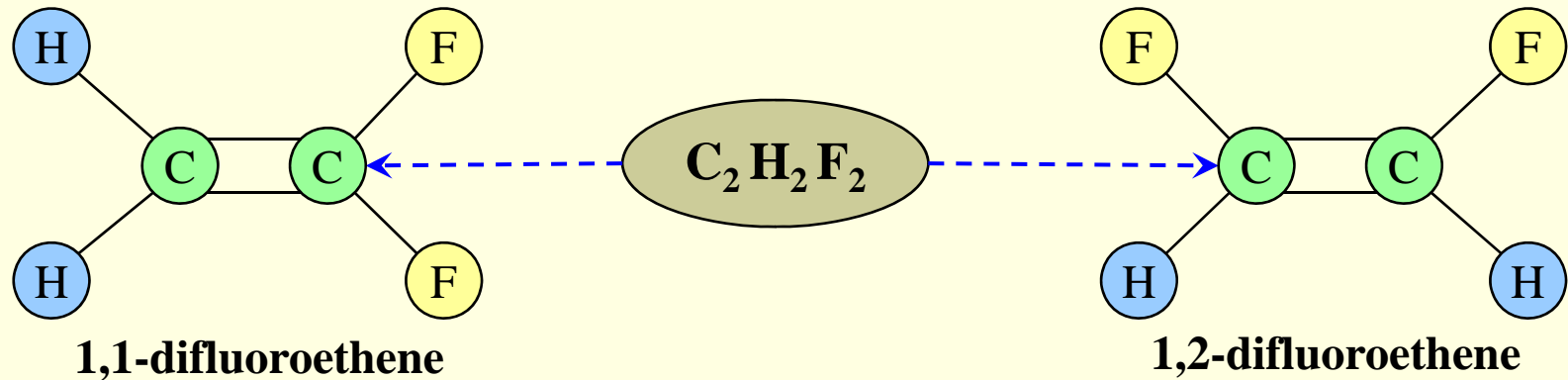
On Canonically Ordering UAV Swarms

- **Motivation:** apply PageRank, as goal is to query data
- **Observation:** random graphs yield canonical order
- **Observation:** which determines graph isomorphism
- **Motivation:** use a larger matrix, e.g., the inverse
- **Observation:** canonically labels *many* more graphs
- **Observation:** is 2-D Weisfeiler-Lehman stabilization
- **Motivation:** find a linear algebraic k -D stabilization
- **Observation:** 1-D is same as equitable partition
- **Observation:** 1-D stabilization done with $\mathbf{1}^n$ vector

Goal: Deciding Graph Isomorphism

Construct a linear algebraic algorithm for finding a k -D Weisfeiler-Lehman (WL) stabilization.

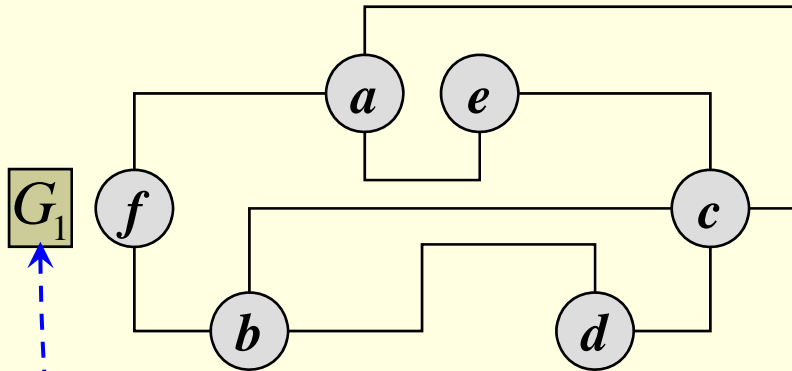
- May yield canonical ordering without search tree
- First linear algebraic k -D WL stabilization method?



- **Classical:** chemical isomers, OCR, CAD/VLSI
- **Military:** find data, detect threats, verify networks

Isomorphism, a Deceptively Simple Idea...

Question: Are G_1 & G_2 or A_1 & A_2 isomorphs?



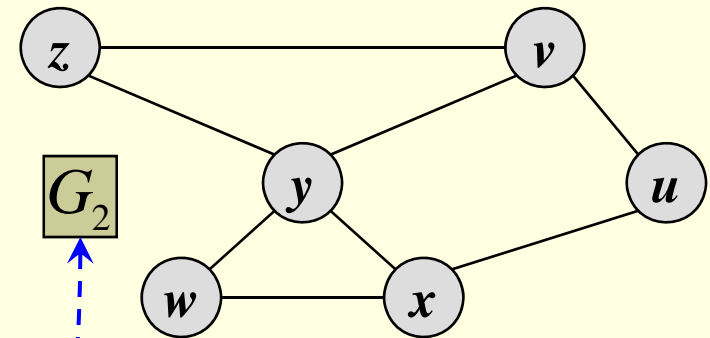
G_1

A_1

	a	b	c	d	e	f
a	0	0	1	0	1	1
b	0	0	1	1	0	1
c	1	1	0	1	1	0
d	0	1	1	0	0	0
e	1	0	1	0	0	0
f	1	1	0	0	0	0

?
 \cong

?
 \cong



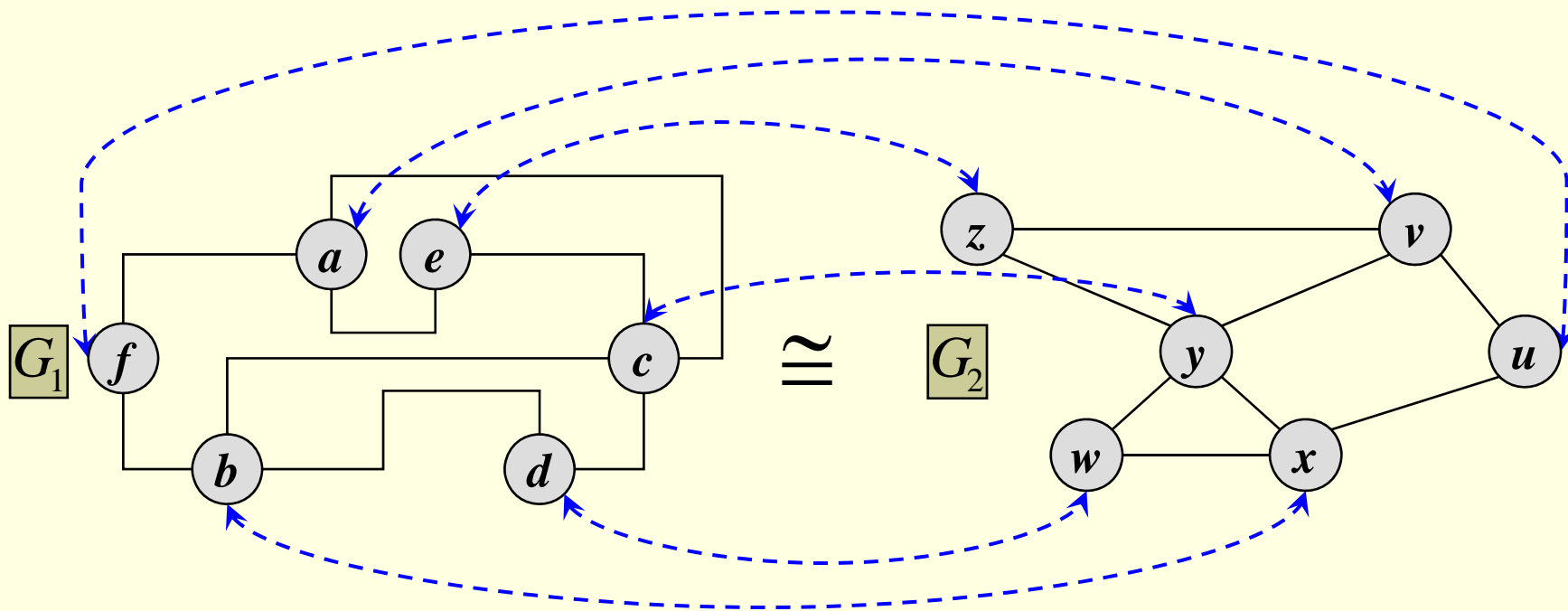
G_2

A_2

	u	v	w	x	y	z
u	0	1	0	1	0	0
v	1	0	0	0	1	1
w	0	0	0	1	1	0
x	1	0	1	0	1	0
y	0	1	1	1	0	1
z	0	1	0	0	1	0

Determining Graph Isomorphism

Answer: $[fadbce] \cong [uvwxyz], \phi = [6, 1, 4, 2, 3, 5]$



$$\exists \phi(V_1) = V_2 \text{ s.t.}$$

$$G_1 \cong G_2 \iff \forall e_i = \{v_a, v_b\} \in E_1 \quad \wedge \quad v_a, v_b \in V_1,$$

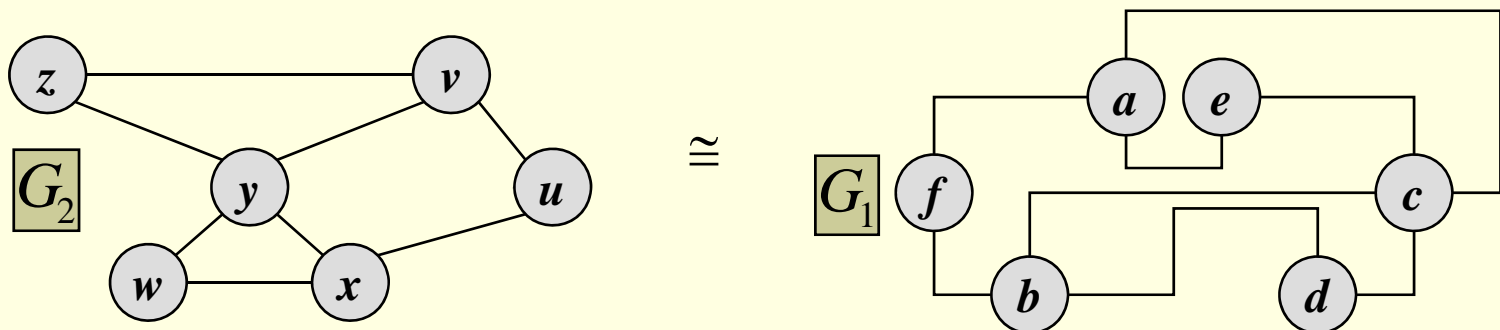
$$\exists e_j = \{\phi(v_a), \phi(v_b)\} \in E_2 \quad \wedge \quad \phi(v_a), \phi(v_b) \in V_2$$

Determining Matrix Isomorphism

■ **Equivalent:** $G_1 \cong G_2 \leftrightarrow A_1 \cong A_2$

■ **Efficient:** $A_2 = P \cdot A_1 \cdot P^{-1} = P \cdot A_1 \cdot P^T, O(n^2)$

							A_2								P								A_1								P^T																				
							u	v	w	x	y	z								1	2	3	4	5	6								a	b	c	d	e	f								6	1	4	2	3	5
u	0	1	0	1	0	0	=	6	0	0	0	0	0	0	1	×	a	0	0	1	0	1	1	×	1	0	1	0	0	0	0	0																			
v	1	0	0	0	1	1		1	1	0	0	0	0	0	0		b	0	0	1	1	0	1		2	0	0	0	1	0	0	0																			
w	0	0	0	1	1	0		4	0	0	0	1	0	0		c	1	1	0	1	1	0		3	0	0	0	0	1	0	0																				
x	1	0	1	0	1	0		2	0	1	0	0	0	0		d	0	1	1	0	0	0		4	0	0	1	0	0	0	0																				
y	0	1	1	1	0	1		3	0	0	1	0	0	0		e	1	0	1	0	0	0		5	0	0	0	0	0	0	1																				
z	0	1	0	0	1	0		5	0	0	0	0	1	0		f	1	1	0	0	0	0		6	1	0	0	0	0	0	0																				



How is Graph Isomorphism Decided?

Assume G is simple & connected

1. Detect non-isomorphism via incomplete invariants

$O(n)$: $|V|$

$O(n^2)$: $|E|$, # of components, degree sequence, ...

$O(n^3)$: eigenvalues, # of triangles, inverse, ...

2. Then compute a

matching between G_1 & G_2 , e.g., VF2

OR

complete invariant, as in *nauty*,

- e.g., the *minimum canonical isomorph* (MCI)
- More useful approach in graph databases

A Template for Deciding Isomorphism

1. **Compare** invariants in increasing complexity order

$$\Psi = \left[|V|, |E|, \text{sort}(\{\text{deg}(v_i), i = 1, 2, \dots, n\}), \text{eig}(\mathbf{A}), \dots \right]$$

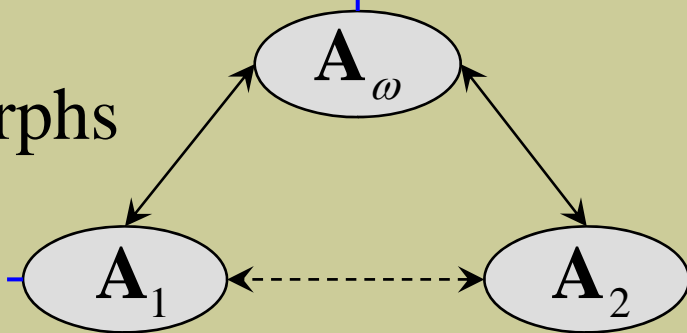
2. **Compute** a canonical isomorph

$$\mathbf{A}_\omega = \mathbf{P}_\omega \cdot \mathbf{A} \cdot \mathbf{P}_\omega^T$$

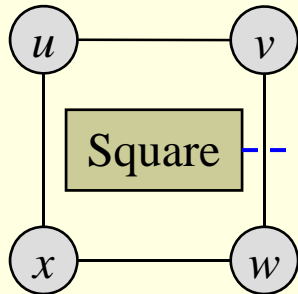
The hard part!

3. **Compare** the canonical isomorphs

$$(\mathbf{A}_1)_\omega \equiv (\mathbf{A}_2)_\omega \iff \mathbf{A}_1 \cong \mathbf{A}_2$$



The MCI, or “All Graphs Have a Number”



Square

Square's (3) Unique Isomorphs

	<i>u</i>	<i>w</i>	<i>x</i>	<i>v</i>
<i>u</i>	0	0	1	1
<i>w</i>	0	0	1	1
<i>x</i>	1	1	0	0
<i>v</i>	1	1	0	0

	<i>u</i>	<i>v</i>	<i>w</i>	<i>x</i>
<i>u</i>	0	1	0	1
<i>v</i>	1	0	1	0
<i>w</i>	0	1	0	1
<i>x</i>	1	0	1	0

	<i>u</i>	<i>x</i>	<i>w</i>	<i>v</i>
<i>u</i>	0	1	1	0
<i>x</i>	1	0	0	1
<i>w</i>	1	0	0	1
<i>v</i>	0	1	1	0

MCI!

Concatenate columns of upper-right triangle

$$011110_2 = 30_{10}$$

$$101101_2 = 45_{10}$$

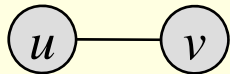
$$110011_2 = 51_{10}$$

I've Got Your Number!



	u
u	0

$$0_2 = 0_{10}$$



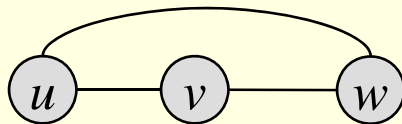
	u	v
u	0	1
v	1	0

$$1_2 = 1_{10}$$



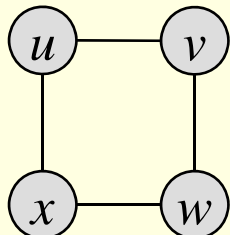
	u	v	w
u	0	0	1
v	0	0	1
w	1	1	0

$$011_2 = 3_{10}$$



	u	v	w
u	0	1	1
v	1	0	1
w	1	1	0

$$111_2 = 7_{10}$$



	u	v	w	x
u	0	0	1	1
v	0	0	1	1
w	1	1	0	0
x	1	1	0	0

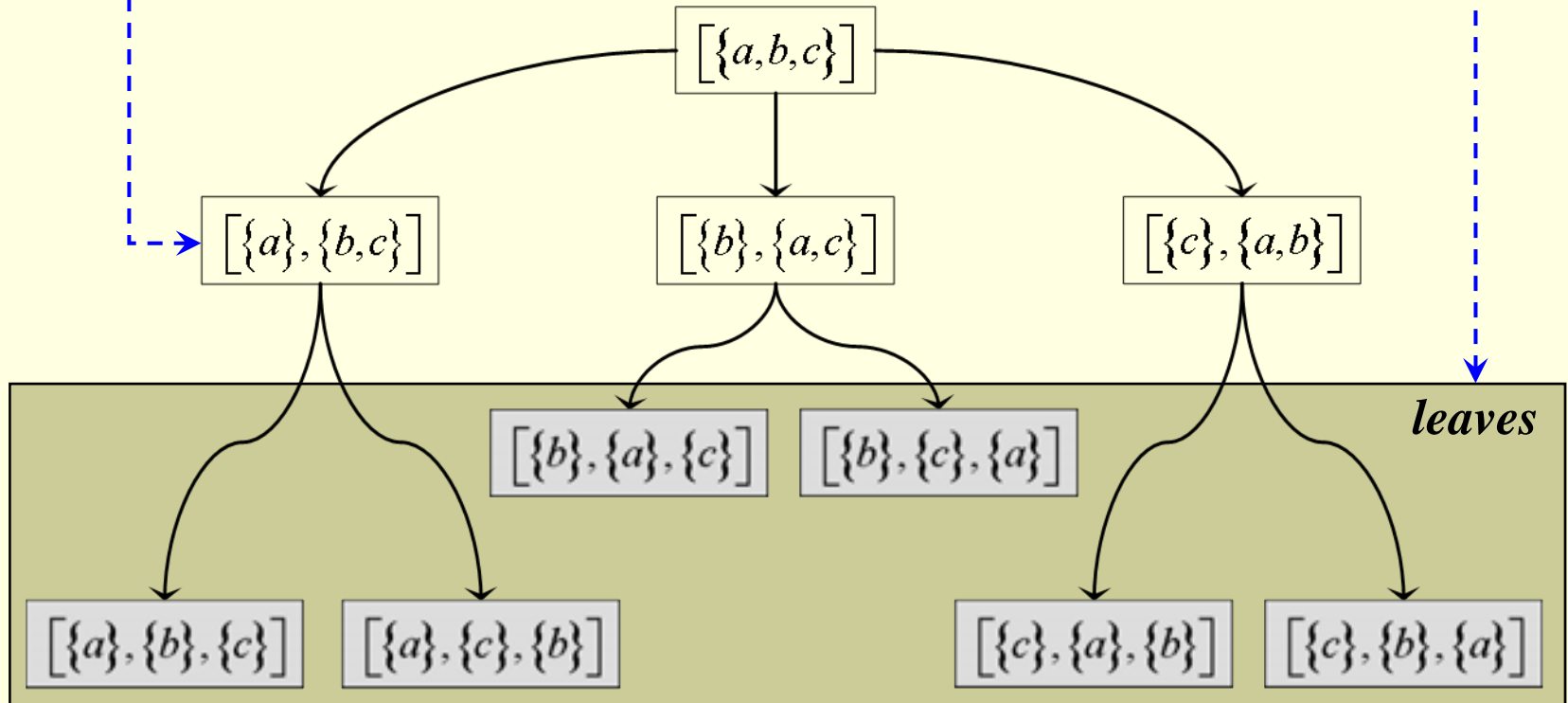
$$011110_2 = 30_{10}$$

Finding the MCI...

Examine $n!$ permutations?

$[abc], [acb], [bac], [bca], [cab], [cba]$

Perform depth-first search on partitions



“The” Question...

The canonization algorithm, *nauty*, prunes the tree via partitions (vertical) & automorphisms (horizontal).

So why not just use *nauty*?

1. The *nauty* copyright — “exception of ... [an] application with *nontrivial* military significance”
2. Require canonical order w.r.t. to vertex importance
3. Further relations b/w graph theory & linear algebra
4. Increase understanding of problem’s complexity
5. Last, but not least, to write a dissertation

k -D Weisfeiler-Lehman Stabilization

■ 1-D: Consider 1-tuples, $O(n^2 \log n)$

Proposed often, Schwenk named *equitable partition*, 1974

Default stabilization method applied in *nauty*

■ 2-D: Consider 2-tuples, $O(n^3 \log n)$

Proposed by Weisfeiler & Lehman (WL), 1968

Coherent (cellular) configuration, Ehrenfeucht-Fraïssé, ...

■ k -D: Consider k -tuples, $O(n^{k+1} \log n)$

Proposed as *deep stabilization* by Weisfeiler, 1976

Shown by Cai, Furer & Immerman (CFI) as insufficient

■ General Comments

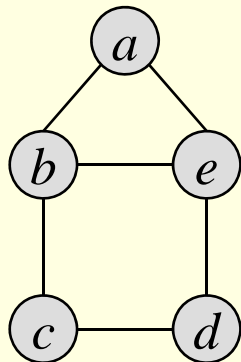
$O(\log n)$ term due to *stabilization*, i.e., *refinement* iterations

May have to apply *individualization*, e.g., on “hard” graphs

What is an Equitable Partition?

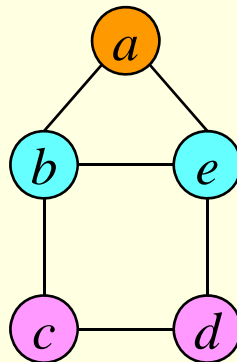
Definition: Given a partition, P , of B blocks (cells), the vertices of a block, b_i , share the same *number* of neighbors in any block, b_j , including their own ($i = j$).

Not Equitable!



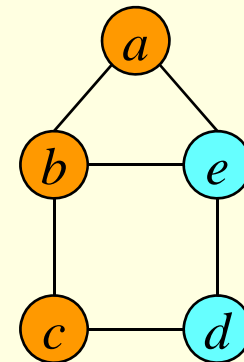
$[\{abcde\}]$

Equitable!



$[\{a\}, \{be\}, \{cd\}]$

Not Equitable!



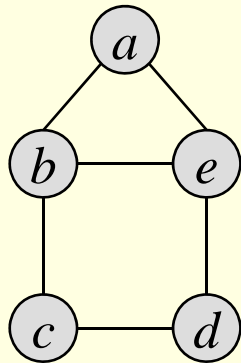
$[\{abc\}, \{de\}]$

House Graph [Gol80]

Common Equitable Partitions

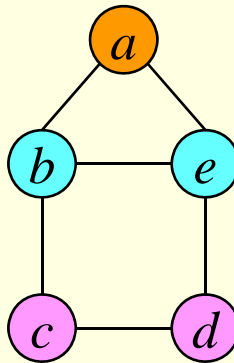
Historical note: equitable partitions are defined using combinatorial tools, but Schwenk's initial application was as divisors of a graph's eigenspectrum.

**Unit
(Root)**



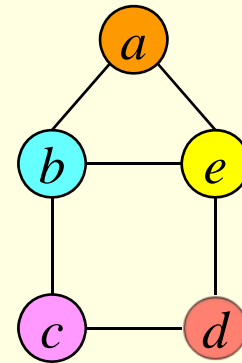
$[\{abcde\}]$

**Automorphism
(Orbits)**



$[\{a\}, \{be\}, \{cd\}]$

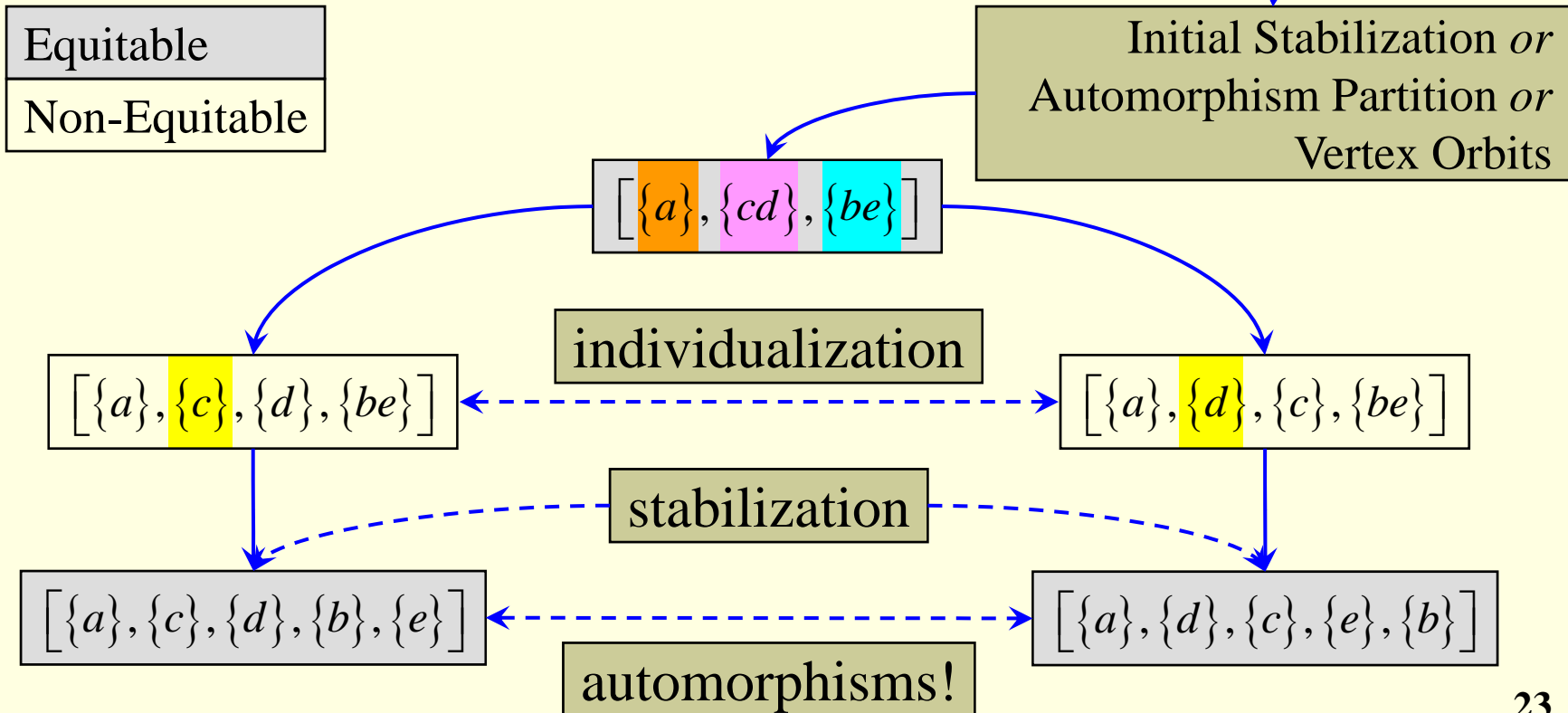
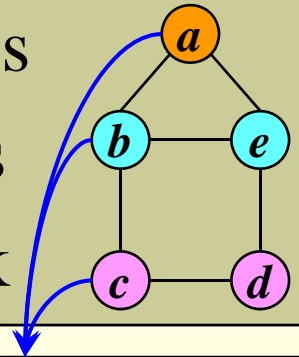
**Leaf
(Permutation)**



$[\{a\}, \{c\}, \{d\}, \{b\}, \{e\}]$

Using Equitable Partitions (*nauty*, 1-D WL)

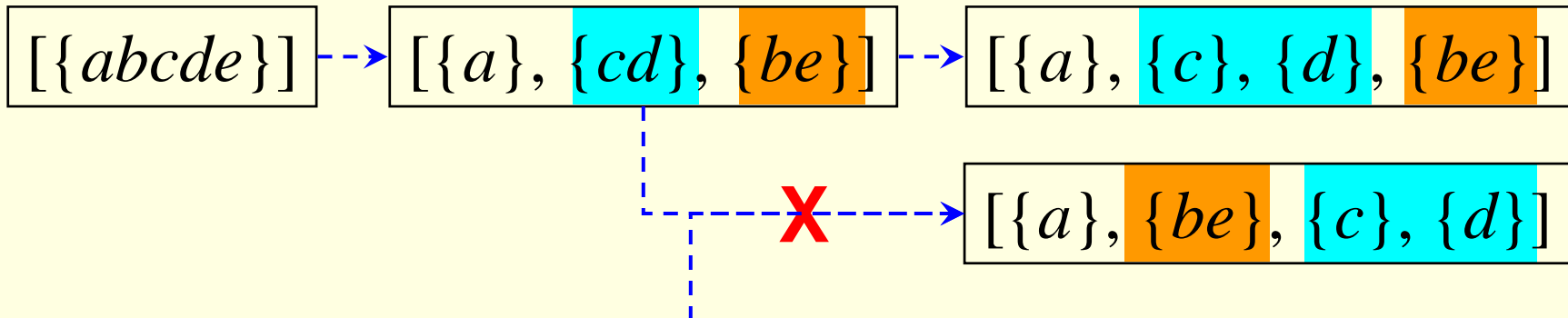
- House graph search tree has $5! = 120$ leaves
- 1-D stabilization yields a tree with 2 leaves
- Eliminates 118/120ths (98.3%) of the work



Partition Conditions...

1. An arbitrary input partition, P
 - is unique coarsest k -D equitable partition *or*
 - yields *finer* coarsest k -D equitable partition

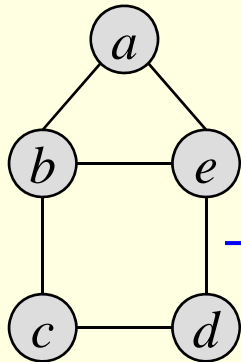
2. A child partition is *ordered* w.r.t. to its parent



$\{cd\}$ & $\{be\}$ violate *block* order of parent

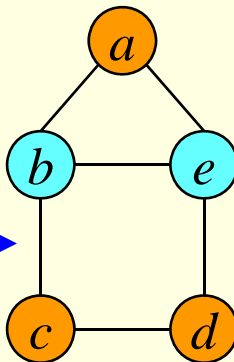
1-D WL Refinement (definition)

Split block if violates the definition, i.e., if 2+ vertices in same block have different number of neighbors in at least one block, e.g., their own block.



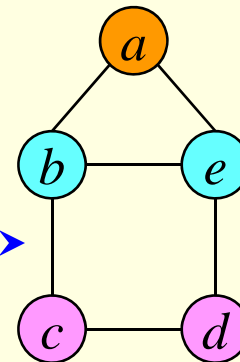
$[\{abcde\}]$

Not Equitable!



$[\{acd\}, \{be\}]$

Not Equitable!



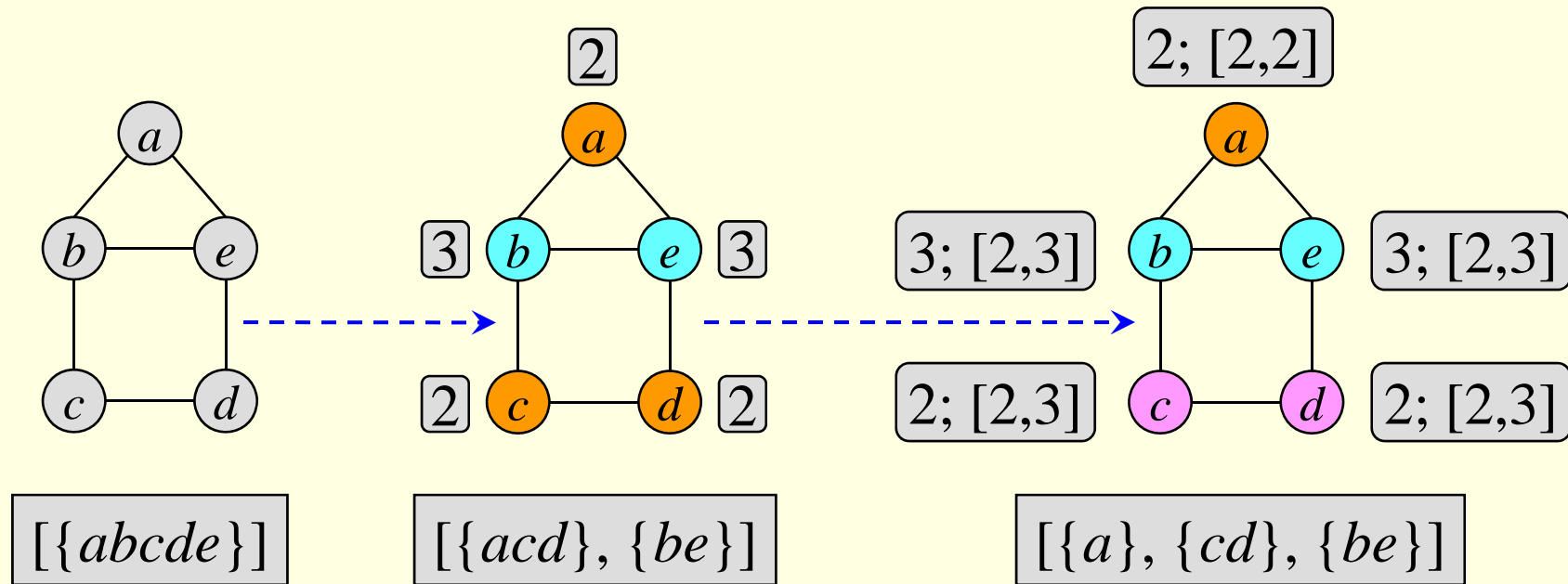
$[\{a\}, \{cd\}, \{be\}]$

Equitable!

refinement is one step, final step is a *stabilization*

1-D WL refinement (Weisfeiler-Lehman)

Label vertices by degree & degrees of neighbors.



Not Equitable!

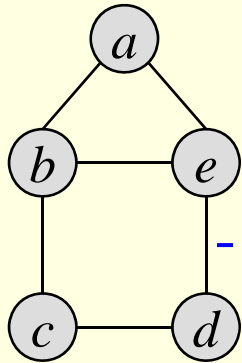
Not Equitable!

Equitable!

intermediate step showing only degree labels

1-D WL Refinement (redux, round #1)

Multiply by *prime* substitutes of degrees and adjacency matrix.



	A				
	a	b	c	d	e
a	0	1	0	0	1
b	1	0	1	0	1
c	0	1	0	1	0
d	0	0	1	0	1
e	1	1	0	1	0

	D				
	a	b	c	d	e
a	2	0	0	0	0
b	0	3	0	0	0
c	0	0	2	0	0
d	0	0	0	2	0
e	0	0	0	0	3

	E				
	a	b	c	d	e
a	1	0	0	0	0
b	0	2	0	0	0
c	0	0	1	0	0
d	0	0	0	1	0
e	0	0	0	0	2

	Q				
	a	b	c	d	e
a	0	2	0	0	2
b	2	0	2	0	4
c	0	2	0	1	0
d	0	0	1	0	2
e	2	4	0	2	0

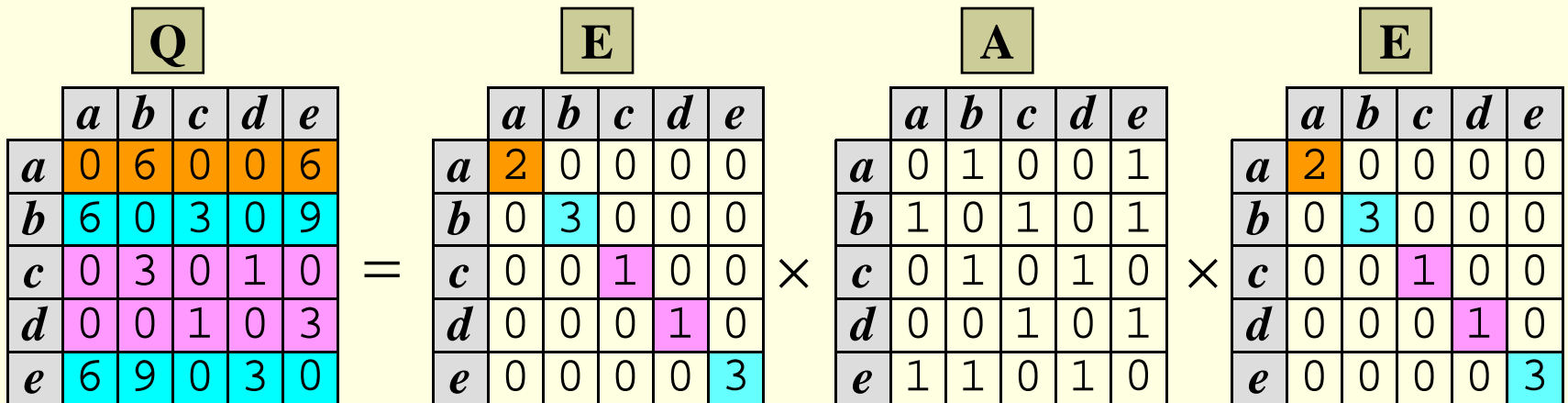
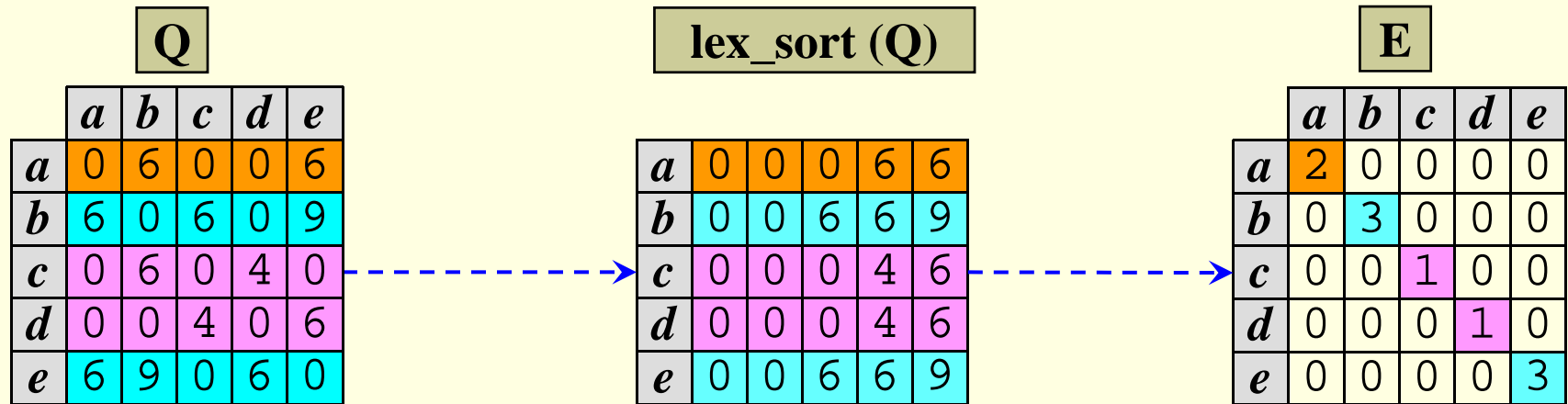
	E				
	a	b	c	d	e
a	1	0	0	0	0
b	0	2	0	0	0
c	0	0	1	0	0
d	0	0	0	1	0
e	0	0	0	0	2

	A				
	a	b	c	d	e
a	0	1	0	0	1
b	1	0	1	0	1
c	0	1	0	1	0
d	0	0	1	0	1
e	1	1	0	1	0

	E				
	a	b	c	d	e
a	1	0	0	0	0
b	0	2	0	0	0
c	0	0	1	0	0
d	0	0	0	1	0
e	0	0	0	0	2

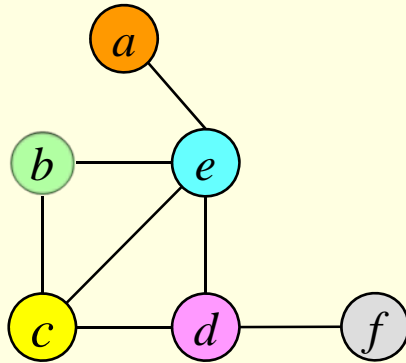
1-D WL Refinement (redux, round #2)

Determine color classes of round and update prime multipliers

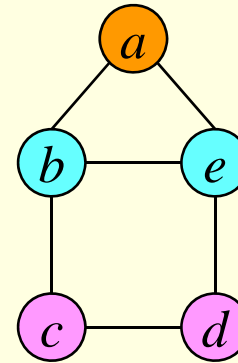


No change in vertex color classes, stabilization achieved!

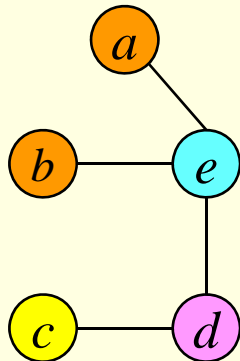
A Few 1-D WL Stabilizations



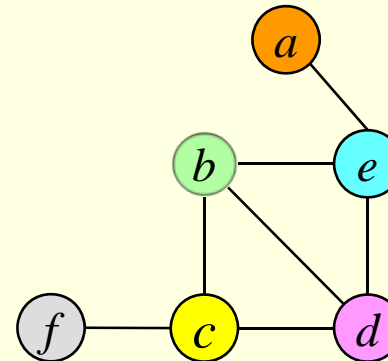
Random Graph
(canonical)



House Graph
(automorphism)



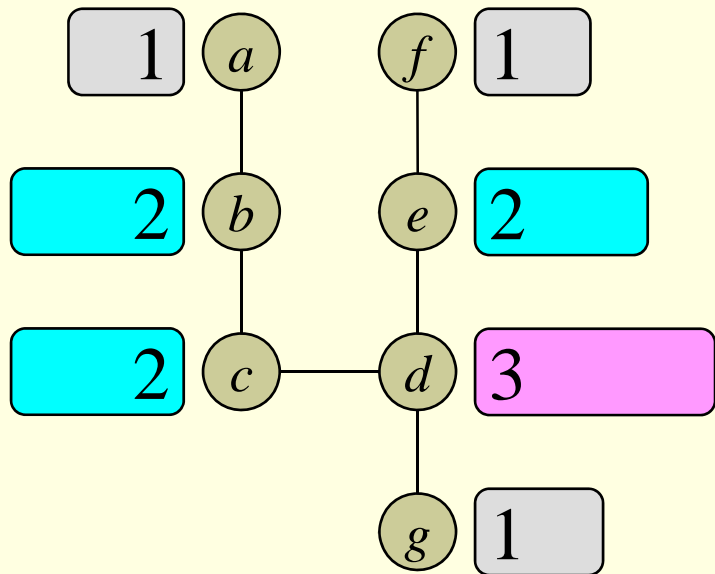
Tree
(automorphism)



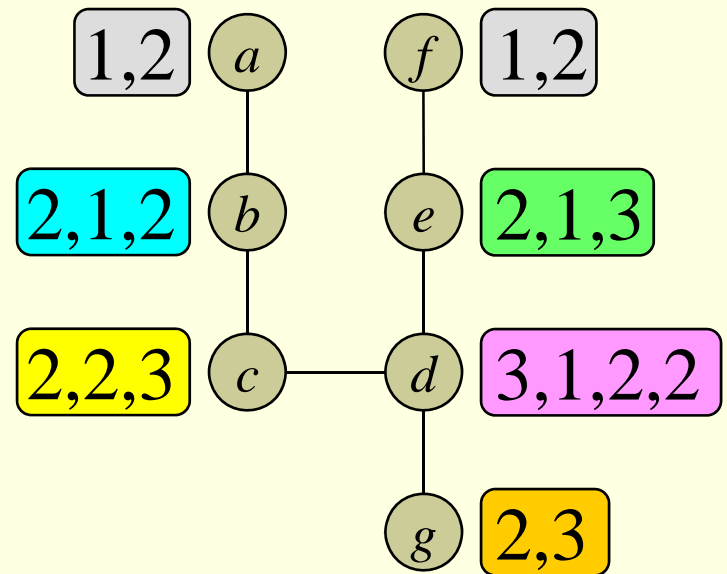
Mansion Graph
(canonical)

1-D WL Stabilization: Refinement #1

tuples (degrees)

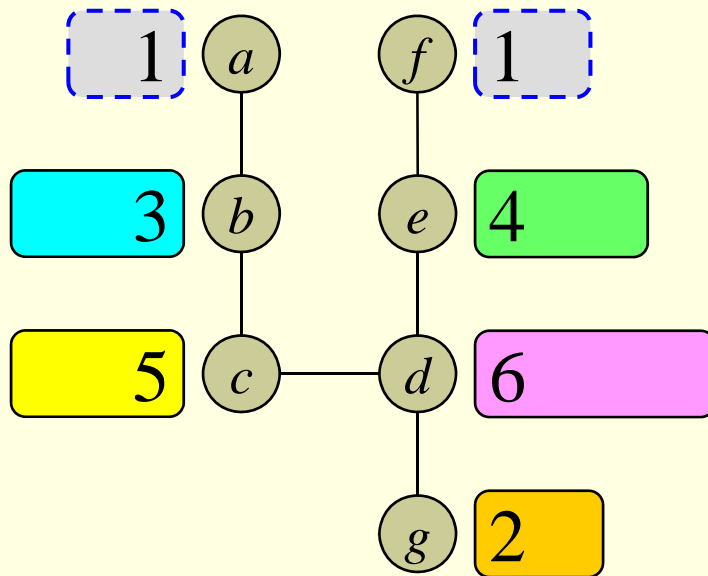


neighbor tuples

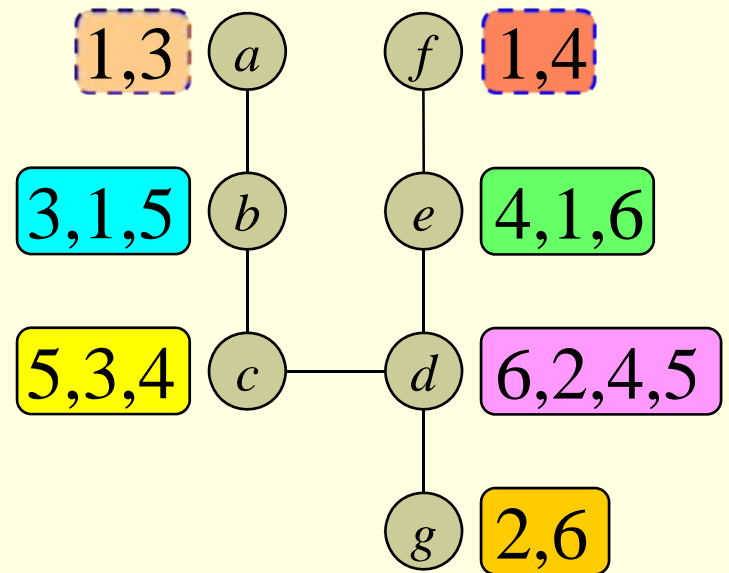


1-D WL Stabilization: Refinement #2

substitute tuples



neighbor tuples

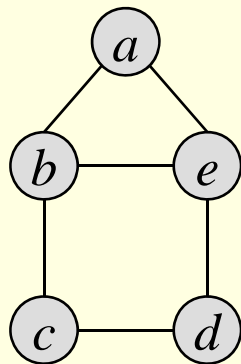


“More” Equitable Partitions

- 1-D WL stabilization & equitable partitions often considered equivalent, *but*
- 2-D WL stabilization yields an equitable partition *and*
- k -D WL stabilization yields an equitable partition.
- However, a $\{2, 3, \dots, k\}$ -D WL stabilization may be more refined relative to a $(k-1)$ -D WL stabilization
- Thus, we may consider it “more” equitable
- The goal, a unit partition of singletons, is the “most” equitable partition that is possible

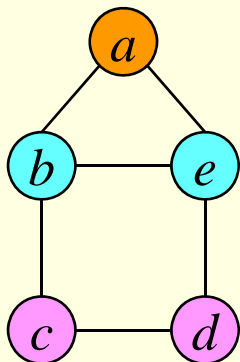
2-D WL Stabilization (*grossly* simplified)

Same idea as 1-D, i.e., is an equitable partition, but w.r.t. to all possible vertex 2-tuples, $O(n^2)$.



	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>
<i>a</i>	2	2, 3	-2, -2	-2, -2	2, 3
<i>b</i>	2, 3	3	2, 3	-2, -3	3, 3
<i>c</i>	-2, -2	2, 3	2	2, 2	-2, -3
<i>d</i>	-2, -2	-2, -3	2, 2	2	2, 3
<i>e</i>	2, 3	3, 3	-2, -3	2, 3	3

**Lexicographical Sorting
(left to right)**

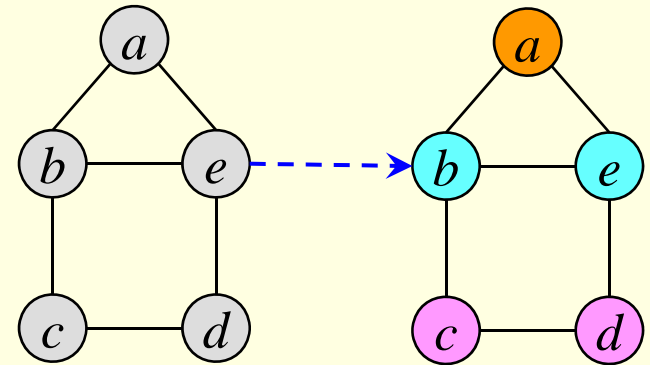


<i>a</i>	0	-2, -2	-2, -2	2, 3	2, 3
<i>b</i>	3	-2, -3	2, 3	2, 3	3, 3
<i>c</i>	2	-2, -3	-2, -2	2, 2	2, 3
<i>d</i>	2	-2, -3	-2, -2	2, 2	2, 3
<i>e</i>	3	-2, -3	2, 3	2, 3	3, 3

2-D WL Stabilization (Weisfeiler-Lehman)

Matrix Multiplication!

- Weisfeiler — polynomials
- Rucker — powers of A
- Here — Rucker + primes



Lex Sort		A^2						A						A							
		<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>
<i>a</i>	1 1 1 1 2	<i>a</i>	2	1	1	1	1	<i>a</i>	0	1	0	0	1	<i>a</i>	0	1	0	0	1		
<i>b</i>	0 1 1 2 3	<i>b</i>	1	3	0	2	1	<i>b</i>	1	0	1	0	1	<i>b</i>	1	0	1	0	1		
<i>c</i>	0 0 1 2 2	<i>c</i>	1	0	2	0	2	<i>c</i>	0	1	0	1	0	<i>c</i>	0	1	0	1	0		
<i>d</i>	0 0 1 2 2	<i>d</i>	1	2	0	2	0	<i>d</i>	0	0	1	0	1	<i>d</i>	0	0	1	0	1		
<i>e</i>	0 1 1 2 3	<i>e</i>	1	1	2	0	3	<i>e</i>	1	1	0	1	0	<i>e</i>	1	1	0	1	0		

This yields the same partition as the less complex 1-D WL stabilization — *why bother?*

k -D WL Refinement (Weisfeiler)

- **Extend** to k -tuples instead of 2-tuples...
- **Miyazaki** showed 1-D stabilization & group methods in *nauty* may yield exponential time on certain graphs
- **Cai, Furer, & Immerman** describe graphs that yield a poly-time canonization algorithm that yield same partitioning, even for $k = n$ that are *not* isomorphs!
- **Weisfeiler** (!) suggests refinement above small values of k , e.g., $k > 3$, may be of limited use, e.g., due to complexity & low probability graph is “difficult”
- **NO** known *linear algebraic* constructions for $k \geq 3$, only discrete k -tuple approaches

Point to Ponder #1

The root equitable partition, or 1-D WL stabilization, yields a *canonical ordering* &/or *uniquely identifies*

- All trees
- All planar graphs
- Random graphs, for $p = 1/2$, *except* for $n^{-1/7}$

Question: for what graphs does 1-D WL stabilization fail to yield any partitioning and/or a canonical form

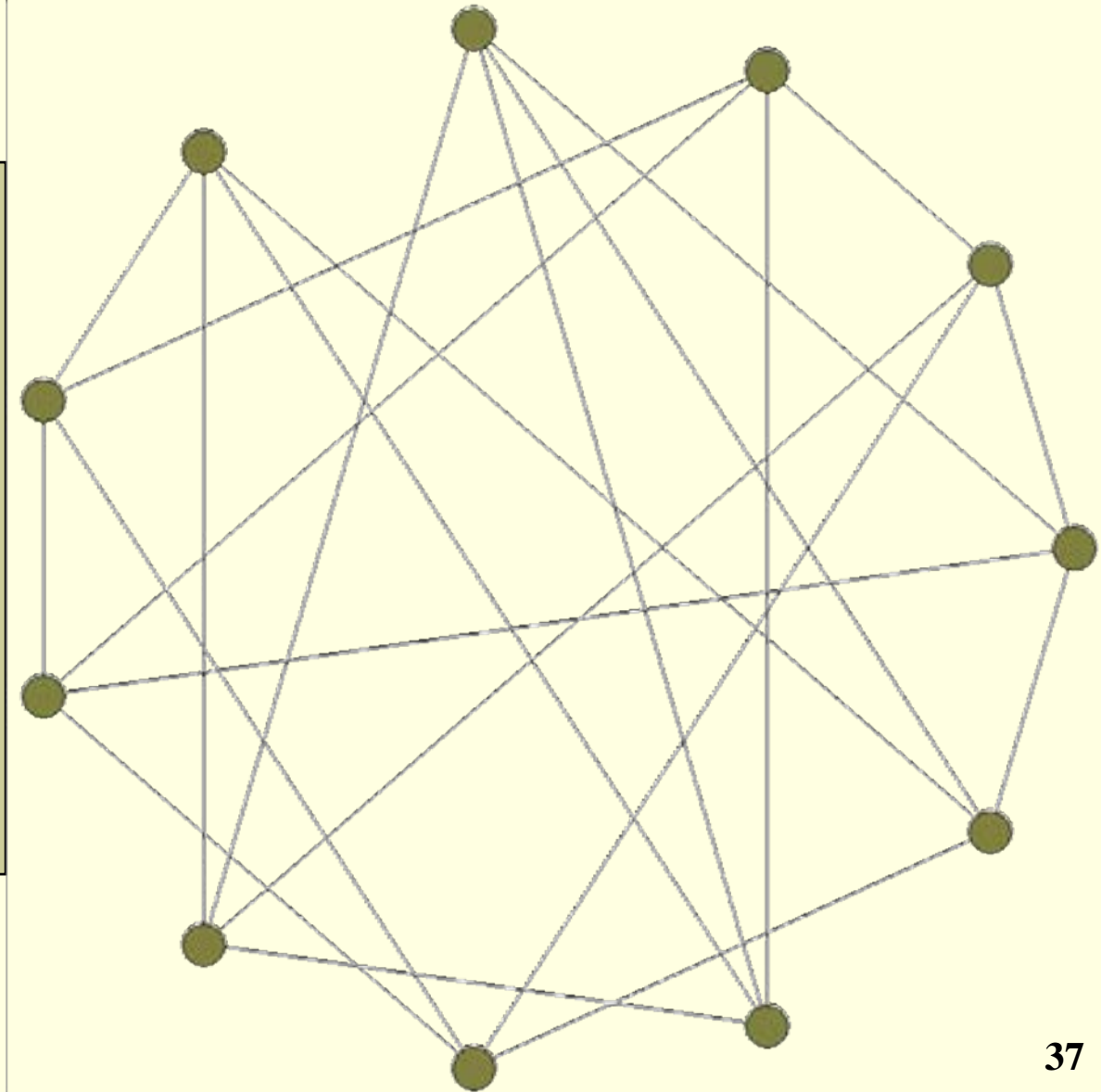
Answer: nearly all regular graphs

“Hard” for 1-D WL Stabilization

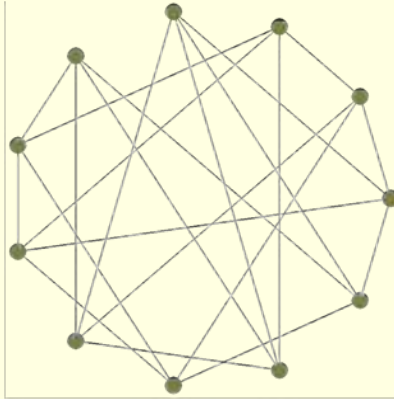
Random Regular,
e.g., $RR(11, 4)$

All vertices are
4-regular

1-D WL yields
no partitioning!



“Easy” for 2-D WL Stabilization

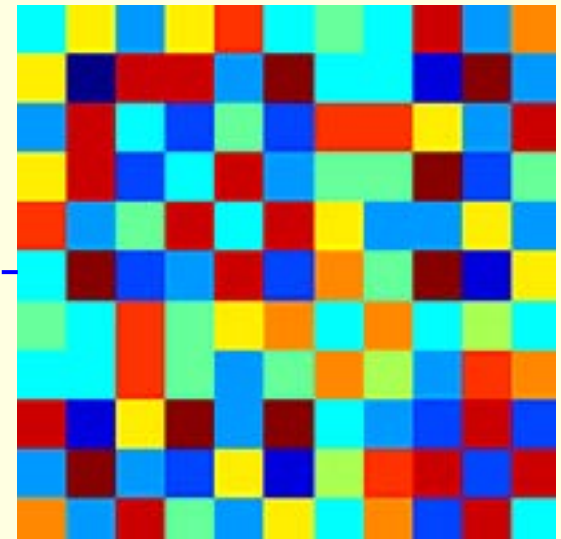
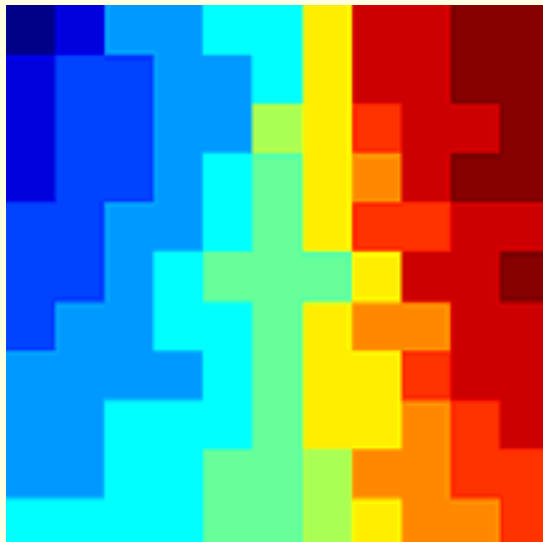


```

0 0 0 1 1 0 0 0 1 0 1
0 0 1 1 0 1 0 0 0 1 0
0 1 0 0 0 0 1 1 0 0 1
1 1 0 0 1 0 0 0 1 0 0
1 0 0 1 0 1 1 0 0 0 0
0 1 0 0 1 0 1 0 1 0 0
0 0 1 0 1 1 0 1 0 0 0
0 0 1 0 0 0 1 0 0 1 1
1 0 0 1 0 1 0 0 0 1 0
0 1 0 0 0 0 0 1 1 0 1
1 0 1 0 0 0 0 1 0 1 0
    
```

Lexicographically Sorted Rows
(all unique, yields canonical order!)

A^3
(prime substitution @ A^2)



Point to Ponder #2

Deciding graph isomorphism is equivalent to

- enumerating all isomorphs
- finding the minimum canonical isomorph
- computing the automorphism partition

Question: when does k -D Weisfeiler-Lehman stabilization yield the automorphism partition

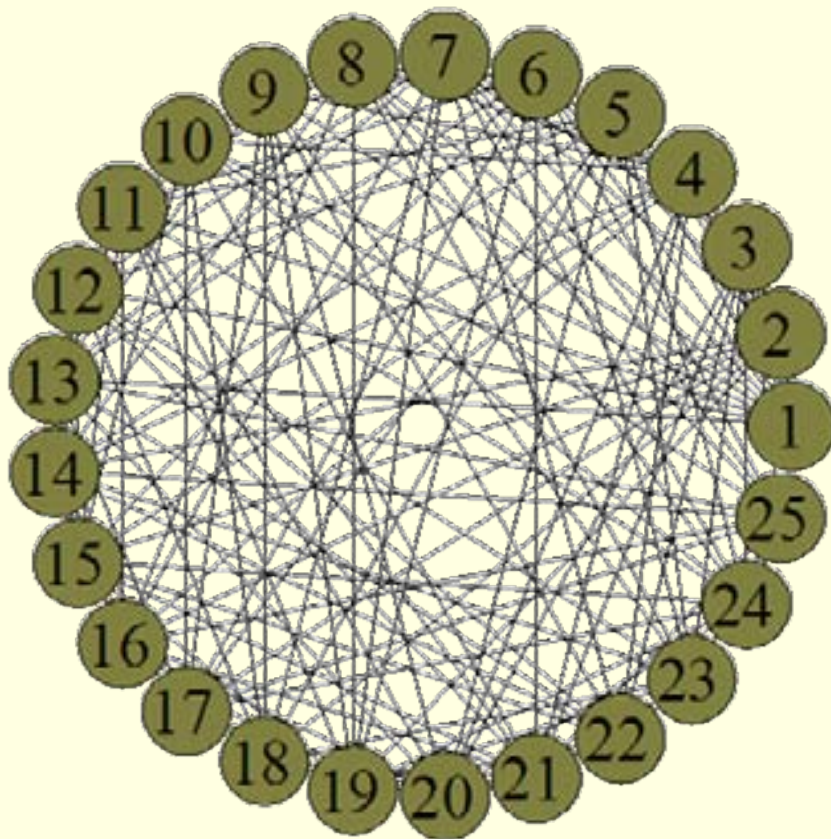
Answer: almost always...but there is no known polynomial test of when this condition is satisfied

“Hard” for $\{1, 2\}$ -D WL Stabilization

- Only yields a *trivial* partition for $k = \{1, 2\}$
- Are difficult to find...very difficult...
- Graphs I’m aware of:
 - **Weisfeiler**: a few graphs on 25 & 26 vertices
 - **Cai-Furer-Immerman**: “ k -D WL fails” graphs
 - Possibly, but I have not checked:
 - **Miyazaki**: “*nauty* is exponential” graphs
 - **Mathon**: “isomorphism algorithm” test graphs

“Easy” for k -D WL Stabilization

i.e., partition yielded by k -D stabilization is *strictly* finer than the partition yielded by $(k-1)$ -D stabilization



Automorphism Partition (orbits) [Wei76]

(1, 2)(4, 7)(8, 18)(9, 19)
(10, 16)(11, 17)(12, 14)
(13, 15)(20, 24)(21, 25)
(3)(5)(21)(22)(23)

k -D WL Stabilization via Linear Algebra

- Linear System, $\mathbf{A} \cdot \mathbf{X} = \mathbf{B}$,
 - \mathbf{A} : adjacency matrix of a graph, G ,
 - \mathbf{B} : matrix of knowns dependent on k ,
 - \mathbf{X} : information matrix,
 - \mathbf{P} : permutation matrix, where
 - $\mathbf{P} \cdot \mathbf{A} \cdot \mathbf{P}^T \leftrightarrow \mathbf{P} \cdot \mathbf{X} \cdot \mathbf{P}^T$
- Gershgorin Circle Theorem
- Laplacian

Graphs & Linear Systems

- Linear System: $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$
- Matrix Inverse: $\mathbf{A} \cdot \mathbf{X} = \mathbf{B} = \mathbf{I}$, i.e., $\mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{I}$
- Do graphs yield solvable linear systems?
 - Given an adjacency matrix, \mathbf{A} , the inverse, \mathbf{A}^{-1} , may *not* exist!
 - Perhaps we could use the *pseudoinverse*
 - $\mathbf{A}^\dagger = (\mathbf{A}^T \cdot \mathbf{A})^{-1} \cdot \mathbf{A}^T$, where
 - if \mathbf{A}^{-1} exists, $\mathbf{A}^{-1} = \mathbf{A}^\dagger$
 - Or, apply *isomorphism-preserving perturbations* to \mathbf{A} such that $(\mathbf{A}')^{-1}$ exists...

Gershgorin Circle Theorem (GCT)

- Establishes each eigenvalue, λ_i , of \mathbf{A} is
 - contained in a disk, \mathbf{d}_i ,
 - centered @ $\mathbf{A}_{i,i}$, and
 - of a radius, $r_i = \sum_{i \neq j} \mathbf{A}_{i,j}$
- If $\mathbf{A}_{i,i} > \sum_{i \neq j} \mathbf{A}_{i,j}$
 - \mathbf{A} is *strictly doubly diagonally dominant* &
 - all eigenvalues are *strictly positive*
 - hence, \mathbf{A} is *positive definite* (by definition, [Str06])
 - therefore, \mathbf{A}^{-1} exists (sufficient condition, [Tau49])

The Laplacian

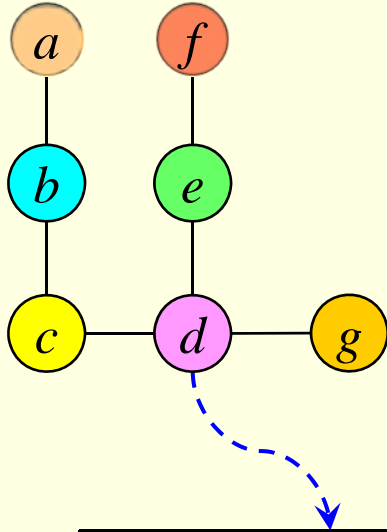
$\mathbf{L} = \mathbf{D} - \mathbf{A}$ Laplacian [BeW04]

$\mathbf{L}^+ = \mathbf{D} + \mathbf{A}$ signless Laplacian [HaS04]

$\mathbf{L}^{+\varepsilon} = \mathbf{D} + \mathbf{A} + \vec{\varepsilon} \cdot \mathbf{I}$ modified Laplacian [AMB+07]

- $\mathbf{L}^{+\varepsilon} = \mathbf{D} + \mathbf{A} + \mathbf{D}^{-1}$ [AMB+07]
- Diagonally dominant ($\mathbf{A}_{i,i} > \sum \mathbf{A}_{i,j}, i \neq j$)
- Positive definite (GCT)
- \mathbf{A}^{-1} exists & is unique up to isomorphism
- \mathbf{A}^{-1} computable w/Cholesky decomposition

1-D WL Stabilization (1s vector, 1 of 3)



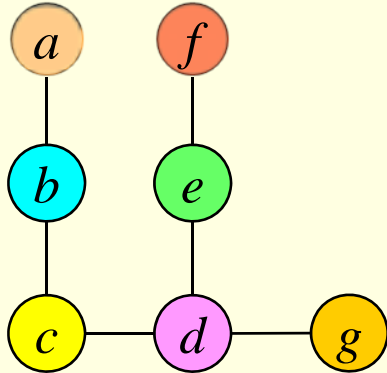
recalling, this tree required (2) *discrete* 1-D WL refinements

$$\mathbf{A}' = \mathbf{A} + \mathbf{D} + \mathbf{D}^{-1}$$

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>
<i>a</i>	0	1	0	0	0	0	0
<i>b</i>	1	0	1	0	0	0	0
<i>c</i>	0	1	0	1	0	0	0
<i>d</i>	0	0	1	0	1	0	1
<i>e</i>	0	0	0	1	0	1	0
<i>f</i>	0	0	0	0	1	0	0
<i>g</i>	0	0	0	1	0	0	0

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>
<i>a</i>	2	1	0	0	0	0	0
<i>b</i>	1	2.5	1	0	0	0	0
<i>c</i>	0	1	2.5	1	0	0	0
<i>d</i>	0	0	1	3.3	1	0	1
<i>e</i>	0	0	0	1	2.5	1	0
<i>f</i>	0	0	0	0	1	2	0
<i>g</i>	0	0	0	1	0	0	2

1-D WL Stabilization (1s vector, 2 of 3)



but this tree only requires (1)
linear algebraic 1-D WL refinement

$$\mathbf{A}' = \mathbf{A} + \mathbf{D} + \mathbf{D}^{-1}$$

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>
<i>a</i>	2	1	0	0	0	0	0
<i>b</i>	1	2.5	1	0	0	0	0
<i>c</i>	0	1	2.5	1	0	0	0
<i>d</i>	0	0	1	3.3	1	0	1
<i>e</i>	0	0	0	1	2.5	1	0
<i>f</i>	0	0	0	0	1	2	0
<i>g</i>	0	0	0	1	0	0	2

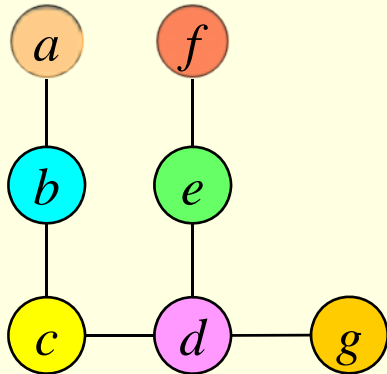
×

<i>x</i>
0.48
0.05
0.41
-0.07
0.28
0.36
0.53

=

<i>b</i>
1
1
1
1
1
1
1

1-D WL Stabilization (1s vector, 3 of 3)



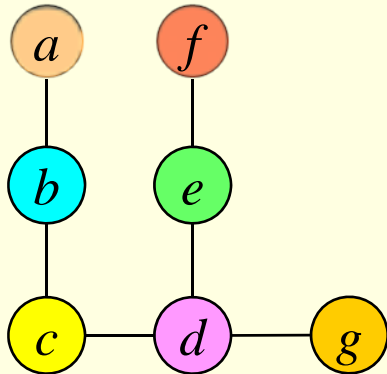
*If more than one iteration is required, A & b are given unique **prime** entries...*

$$A' = A + D + D^{-1}$$

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>
<i>a</i>	39.03	3.13	0	0	0	0	0
<i>b</i>	3.13	72.01	3.11	0	0	0	0
<i>c</i>	0	3.11	55.02	2.11	0	0	0
<i>d</i>	0	0	2.11	66.02	2.5	0	2.17
<i>e</i>	0	0	0	2.5	45.02	5.7	0
<i>f</i>	0	0	0	0	5.7	35.03	0
<i>g</i>	0	0	0	2.17	0	0	34.03

	x		b
	?		13
	?		3
	?		11
×	?	=	2
	?		5
	?		7
	?		17

1-D WL Stabilization (vector sum || degree)



Jump-start using vector sum, i.e., the vertex degree vector?

$$\mathbf{A}' = \mathbf{A} + \mathbf{D} + \mathbf{D}^{-1}$$

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>
<i>a</i>	2	1	0	0	0	0	0
<i>b</i>	1	2.5	1	0	0	0	0
<i>c</i>	0	1	2.5	1	0	0	0
<i>d</i>	0	0	1	3.3	1	0	1
<i>e</i>	0	0	0	1	2.5	1	0
<i>f</i>	0	0	0	0	1	2	0
<i>g</i>	0	0	0	1	0	0	2

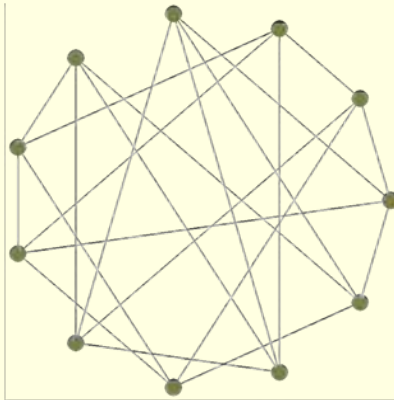
×

<i>x</i>
0.20
0.59
0.31
0.62
0.44
0.28
0.19

=

<i>b</i>
1
2
2
3
2
1
1

2-D WL Stabilization (inverse, 1 of 2)

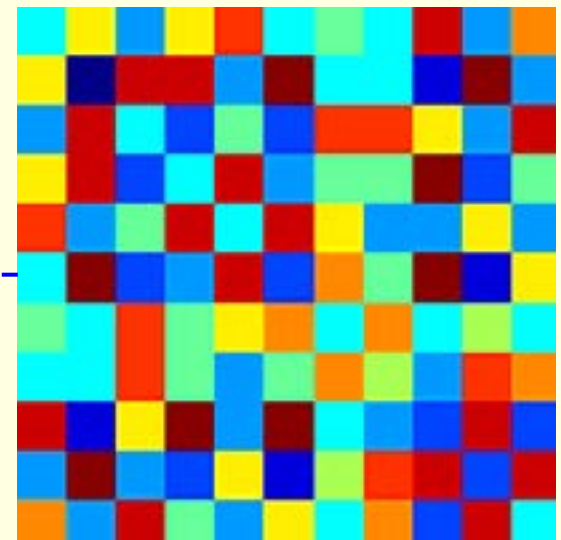
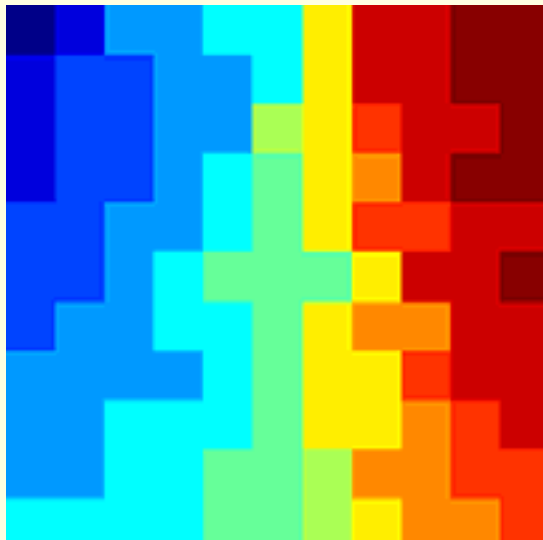


```

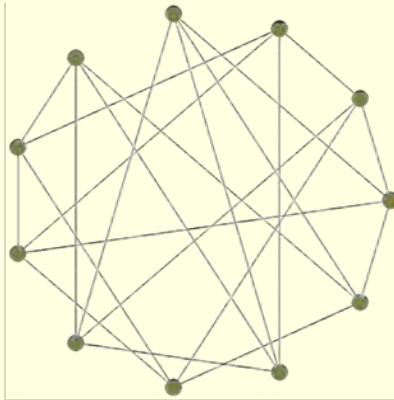
0 0 0 1 1 0 0 0 1 0 1
0 0 1 1 0 1 0 0 0 1 0
0 1 0 0 0 0 1 1 0 0 1
1 1 0 0 1 0 0 0 1 0 0
1 0 0 1 0 1 1 0 0 0 0
0 1 0 0 1 0 1 0 1 0 0
0 0 1 0 1 1 0 1 0 0 0
0 0 1 0 0 0 1 0 0 1 1
1 0 0 1 0 1 0 0 0 1 0
0 1 0 0 0 0 0 1 1 0 1
1 0 1 0 0 0 0 1 0 1 0
    
```

**Lexicographically Sorted Rows
(all unique, yields canonical order!)**

***recalling, A^3*
(prime substitution @ A^2)**



2-D WL Stabilization (inverse, 2 of 2)

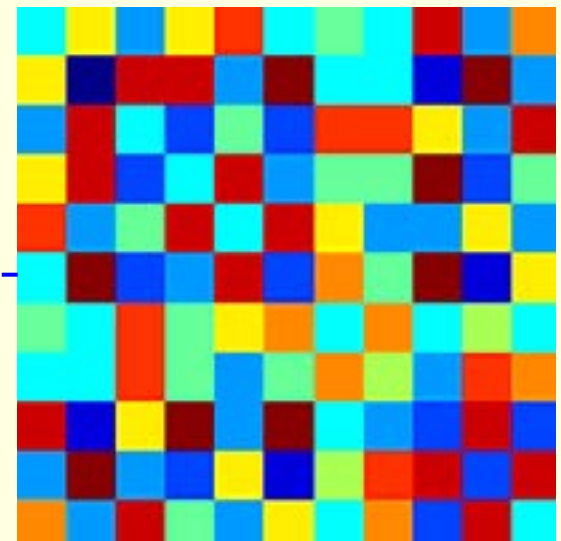
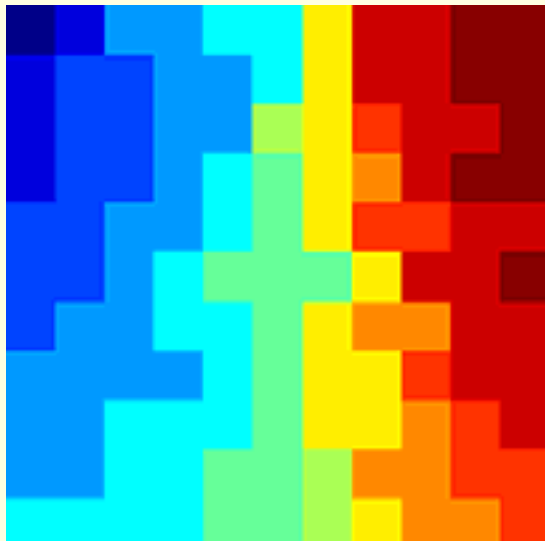


```

0 0 0 1 1 0 0 0 1 0 1
0 0 1 1 0 1 0 0 0 1 0
0 1 0 0 0 0 1 1 0 0 1
1 1 0 0 1 0 0 0 1 0 0
1 0 0 1 0 1 1 0 0 0 0
0 1 0 0 1 0 1 0 1 0 0
0 0 1 0 1 1 0 1 0 0 0
0 0 1 0 0 0 1 0 0 1 1
1 0 0 1 0 1 0 0 0 1 0
0 1 0 0 0 0 0 1 1 0 1
1 0 1 0 0 0 0 1 0 1 0
    
```

**Lexicographically Sorted Rows
(all unique, yields canonical order!)**

$$\left(\mathbf{A} + \mathbf{D} + \mathbf{D}^{-1} \right) \cdot \mathbf{X} = \mathbf{I}^{n,n}$$



2-D WL Stabilization (vector sum – $\mathbf{D} \parallel \mathbf{A}$)

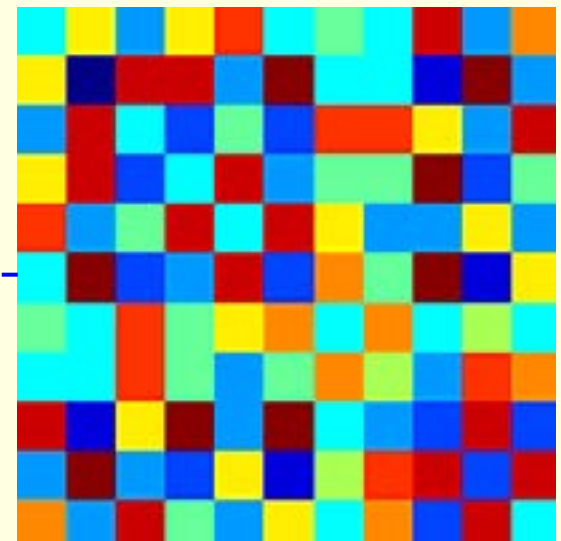
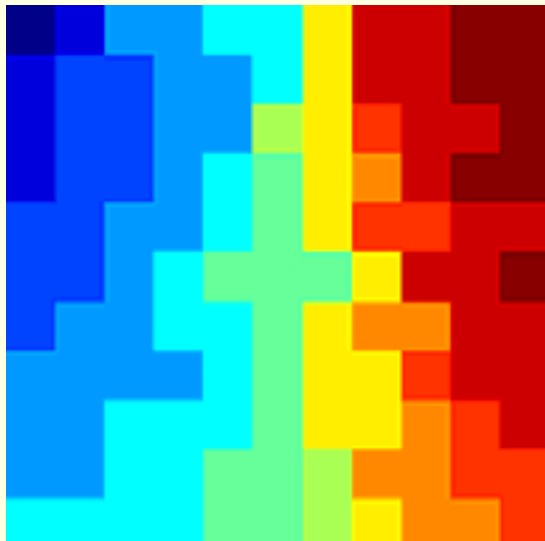


```

0 0 0 1 1 0 0 0 1 0 1
0 0 1 1 0 1 0 0 0 1 0
0 1 0 0 0 0 1 1 0 0 1
1 1 0 0 1 0 0 0 1 0 0
1 0 0 1 0 1 1 0 0 0 0
0 1 0 0 1 0 1 0 1 0 0
0 0 1 0 1 1 0 1 0 0 0
0 0 1 0 0 0 1 0 0 1 1
1 0 0 1 0 1 0 0 0 1 0
0 1 0 0 0 0 0 1 1 0 1
1 0 1 0 0 0 0 1 0 1 0
    
```

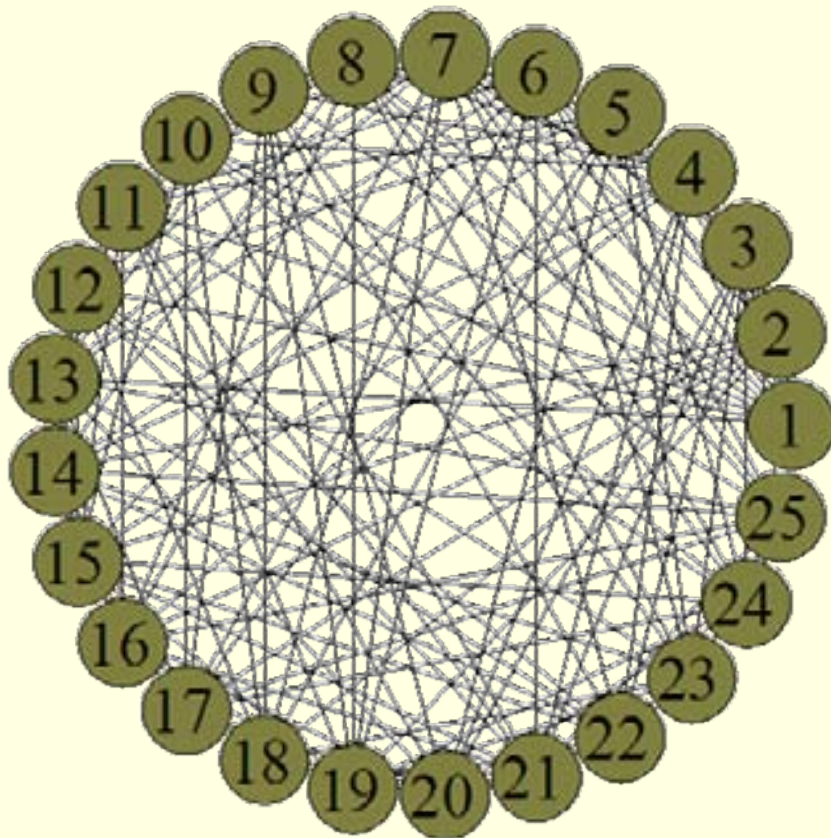
**Lexicographically Sorted Rows
(all unique, yields canonical order!)**

$$\left(\mathbf{A} + \mathbf{D} + \mathbf{D}^{-1} \right) \cdot \mathbf{X} = \mathbf{A} \text{ or } \mathbf{D}$$



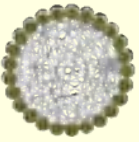
k -D WL Stabilization (naïve, 1 of 3)

recalling, this graph first yields a non-trivial partition
@ $k = 3$



Automorphism Partition (orbits) [Wei76]

$(1, 2)(4, 7)(8, 18)(9, 19)$
 $(10, 16)(11, 17)(12, 14)$
 $(13, 15)(20, 24)(21, 25)$
 $(3)(5)(21)(22)(23)$



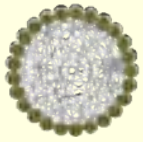
Linear Algebraic Motivation

1. 1-D: $\mathbf{A}' \cdot \mathbf{x} = \mathbf{1}^{n,1}$

2. 2-D: $\mathbf{A}' \cdot \mathbf{X} = \mathbf{I}^{n,n}$

3. k -D: $\mathbf{A}' \cdot \mathbf{X} = \begin{pmatrix} n \\ k \end{pmatrix}$

$$\mathbf{A}' \cdot \mathbf{X} = \begin{bmatrix} 1 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 1 \end{bmatrix}$$



Linear Algebraic Motivation

But it doesn't work 😞!

1. 1-D: $\mathbf{A}' \cdot \mathbf{x} = \mathbf{1}^{n,1}$

2. 0-D: $\mathbf{A}' \cdot \mathbf{x} = \mathbf{I}^n$

3. k -D: $\mathbf{A}' \cdot \mathbf{X} = \begin{pmatrix} n \\ k \end{pmatrix}$

$$\mathbf{A}' \cdot \mathbf{X} = \begin{bmatrix} 1 & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 1 \end{bmatrix}$$

k -D WL Stabilization (vector sum of \mathbf{A})

Linear Algebraic Motivation

\mathbf{A}

1. $\mathbf{A}' \cdot \mathbf{X} = \begin{pmatrix} n \\ k \end{pmatrix}$

	u	v	w	x
u	0	1	0	1
v	1	0	0	0
w	0	0	0	1
x	1	0	1	0

2. Vector sums of \mathbf{A} w.r.t. $\begin{pmatrix} n \\ k \end{pmatrix}$

$$\mathbf{A}' \cdot \mathbf{X} = \begin{bmatrix} 1 & 1 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 1 \\ 0 & 0 & \dots & 1 \end{bmatrix} \rightarrow \begin{bmatrix} uv & uw & ux & vw & vx & wx \\ 1 & 0 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 2 & 1 & 1 & 0 & 1 \end{bmatrix}$$

k -D WL Stabilization (vector multiply of \mathbf{A})

Linear Algebraic Motivation

\mathbf{A}

1. $\mathbf{A}' \cdot \mathbf{X} = \begin{pmatrix} n \\ k \end{pmatrix}$

	u	v	w	x
u	0	1	0	1
v	1	0	0	0
w	0	0	0	1
x	1	0	1	0

2. Vector products of \mathbf{A} w.r.t. $\begin{pmatrix} n \\ k \end{pmatrix}$

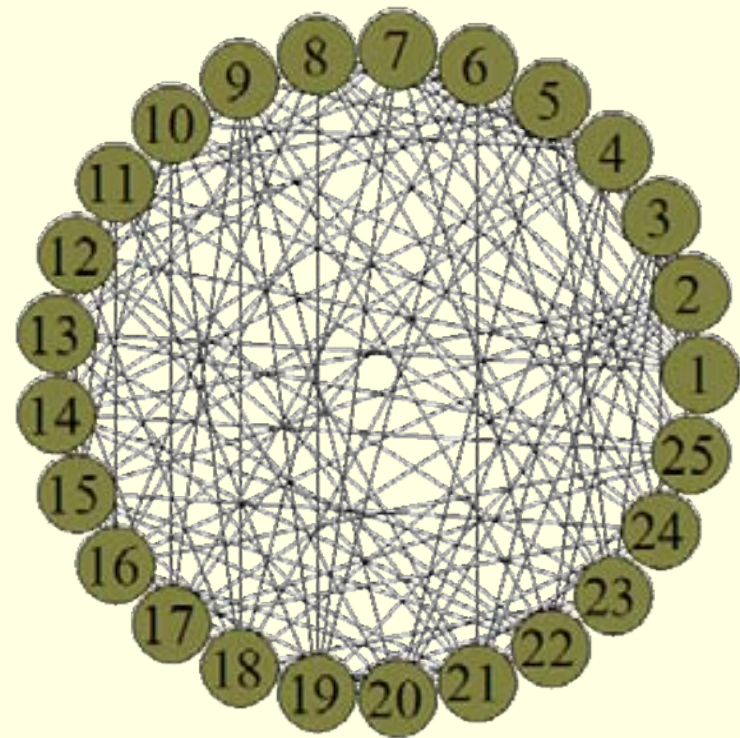
$$\mathbf{A}' \cdot \mathbf{X} = \begin{bmatrix} 1 & 1 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 1 \\ 0 & 0 & \dots & 1 \end{bmatrix} \rightarrow \begin{bmatrix} uv & uw & ux & vw & vx & wx \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

k -D WL Stabilization (add, multiply, ...)

Lexico_Sort(X)
(as we have been)

u	-1.00	0.00	0.17	0.33	0.50	1.50
v	-0.75	-0.25	-0.17	0.42	0.50	1.00
w	-0.75	-0.25	-0.17	0.42	0.50	1.00
x	-1.00	0.00	0.17	0.33	0.50	1.50

Which suffices on
Weisfeiler's graphs, e.g.,



k -D WL Stabilization (a few optimizations)

■ Efficiency

1. Leverage parallel linear algebra libraries
2. Remove singletons from \mathbf{B} , esp. from (n, k)
3. Only update non-singletons in \mathbf{A}

■ Conditioning

1. Prime substitution
2. Logarithm of primes for large n
3. Rounding to accommodate numerical error

Summary

■ *k*-D Weisfeiler-Lehman Stabilization via

- Discrete *k*-tuples modern view
- Matrix multiplication Weisfeiler-Lehman
- Linear systems our work
- Complexity $O(n^{k+1} \cdot \log n)$?

■ Next step

- Discrete or algebraic?
- Formal proof?