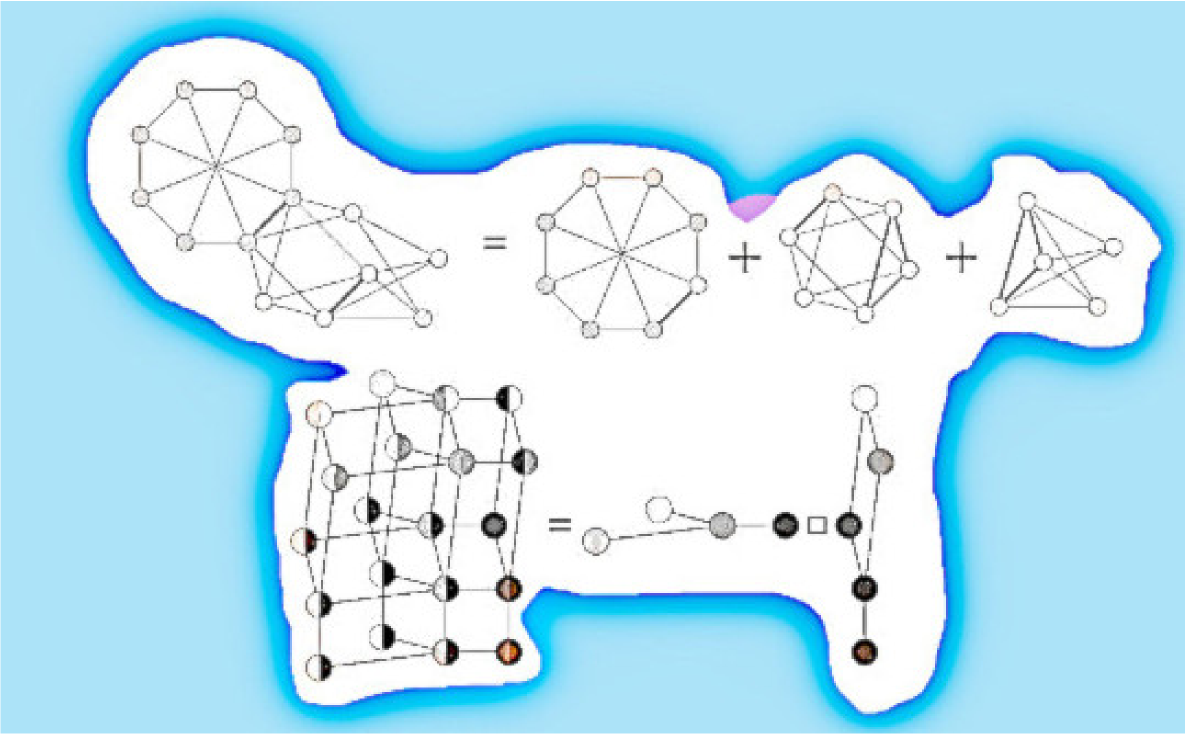
Graph Algebras



Pavel Loskot
pavelloskot@intl.zju.edu.cn

ZJU-UIUC INSTITUTE
Zhejiang University-University of Illinois at Urbana-Champaign Institute
浙江大学伊利诺伊大学厄巴纳香槟校区联合学院

The Tenth International Conference on Advances in Signal, Image and Video Processing (SIGNAL 2025)

March 09 – March 13, 2025, Lisbon, Portugal

About Me

Pavel Loskot joined the ZJU-UIUC Institute as Associate Professor in January 2021. He received his PhD degree in Wireless Communications from the University of Alberta in Canada, and the MSc and BSc degrees in Radioelectronics and Biomedical Electronics, respectively, from the Czech Technical University of Prague. He is the Senior Member of the IEEE, Fellow of the HEA in the UK, and the Recognized Research Supervisor of the UKCGE. In autumn 2024, he was elected the IARIA 2025 Fellow.

In the past nearly 30 years, he was involved in numerous industrial and academic collaborative projects in the Czech Republic, Finland, Canada, the UK, Turkey, and in China. These projects concerned wireless and optical telecommunication networks, and also genetic regulatory circuits, air transport services and renewable energy systems. This experience allowed him to truly understand the interdisciplinary workings, and crossing the disciplines boundaries.

His current research focuses on mathematical and probabilistic modeling, statistical signal processing and classical machine learning for multi-sensor data in biomedicine, computational molecular biology, and wireless communications.

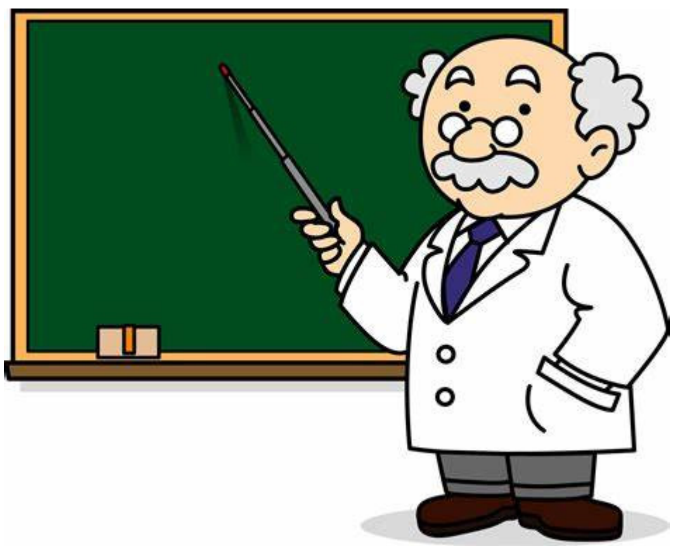
Objectives

Explore modeling abstractions

- algebraic manipulations of mathematical objects
  → assume graphs as nearly implicitly or explicitly ubiquitous objects
- move from numerical (quantitative) to **semantic computations**
- define dynamic systems involving graph sums and products
  → auto-regressive modeling

Topics

1. Algebraic structures and algebras
2. Graph sums, products, and rewriting
3. Algebraic graph theory and autoregressive graph modeling



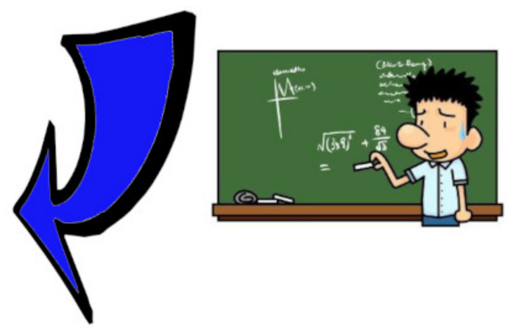
Mathematics-Engineering Gap

Engineering

- pragmatic, design oriented
- things driven
- complexity becoming an issue
- increase use of math models
- increase use of abstractions
- large pool of engineers

Mathematics

- pure vs. applied, but always rigorous
- concepts driven
- study of abstractions
- specialized skills/knowledge
- favorite areas: bio-med, finance
- small pools of mathematicians



Opportunity

- adopt common/advanced math concepts for “easy” use in engineering
- go beyond calculus and numerical computations
- allow working with advanced math objects, structures and models

Focus of This Talk

Informally

- algebras with arithmetic rules used extensively in numerical computations
  $\rightarrow$ arithmetic operations involve numbers representing numerical values
- can explore arithmetic rules for semantic manipulations of math objects
  $\rightarrow$ capture the structure and geometric shapes

Algebras

- define the laws of computation for numbers (number systems)

Abstract algebras

- manipulate algebraic (e.g. numeric and geometric) objects

Specific goals

- assume models where scalar values are replaced with graphs
- explore abstract algebraic operations involving graphs
  $\rightarrow$ graph sums and products
  $\rightarrow$ graph transformations
- eventually
  $\rightarrow$ generalize these concepts to other mathematical objects
  $\rightarrow$ assume mixture models combining math structures and numerical values

Basic Algebraic Structures

Semi-group $(S,+)$

- closed and associative w.r.t. operator '+'

Monoid $(S,+)$

- a semi-group with neutral element, i.e., $a+z=a\ \forall a\in S$

Group $(S,+)$

- a monoid with inverse element, i.e., $a+\bar{a}=z\ \forall a\in S$

Abelian group $(S,+)$

- commutative w.r.t. operator '+', i.e., $a+b=b+a$

Ring $(S,+,*)$

- $(S,+)$ is commutative group and $(S,*)$ is semi-group
- distributive, i.e., $a*(b+c)=a*b+a*c\ \forall a,b,c\in S$

Field $(S,+,*)$

- ring with $(S\setminus\{0\},*)$ being a group

Basic Algebraic Structures (cont.)

Examples

- $(\mathbb{Z}, -)$ is not semi-group (not associative)
- $(\mathbb{N}, +)$ is semi-group (not group, since no 0)
- $(\mathbb{N}_0, +)$ is monoid (no inverse element)
- $(\mathbb{Z}, *)$ is monoid (no inverse element)
- $(\mathbb{Z}, +)$ is group
- $(\mathbb{Z}_n, +)$ is group
- $(\mathbb{Z}_n, *)$ is monoid (no inverse element for 0)
- $(\mathbb{Z}_n \setminus \{0\}, *)$ is group if n is prime
- $(\mathbb{Z}, +, *)$ is ring (not field)
- $(\mathbb{Z}_n, +, *)$ is finite ring and finite field if n is prime

Galois field $\mathrm{GF}(p^k)$

- modulo p arithmetic
- non-zero elements of $\mathrm{GF}(p^k)$ form multiplicative cyclic group
- $\mathrm{GF}(2^2) = \{0, 1, \alpha, 1+\alpha\}$

Addition $x + y$

$x \backslash y$	0	1	α	$1+\alpha$
0	0	1	α	$1+\alpha$
1	1	0	$1+\alpha$	α
α	α	$1+\alpha$	0	1
$1+\alpha$	$1+\alpha$	α	1	0

Multiplication $x \cdot y$

$x \backslash y$	0	1	α	$1+\alpha$
0	0	0	0	0
1	0	1	α	$1+\alpha$
α	0	α	$1+\alpha$	1
$1+\alpha$	0	$1+\alpha$	1	α

Division $x \div y$

$x \backslash y$	1	α	$1+\alpha$
0	0	0	0
1	1	$1+\alpha$	α
α	α	1	$1+\alpha$
$1+\alpha$	$1+\alpha$	α	1

More on Algebraic Structures

Groups

- homomorphism from $(G, \cdot)$ to $(H, *)$ is $\phi : \; G \mapsto H$

$$\phi(g_1 \cdot g_2) = \phi(g_1) * \phi(g_2) \quad \forall g_1, g_2 \in G$$

- the sum $(g_1 + g_2)^p = g_1^p + g_2^p$ for $\forall g_1, g_2 \in \mathrm{GF}(p)$
- special groups
  $\rightarrow$ cosets, finite cyclic groups, abelian groups

Rings

- generalizes fields
  $\rightarrow$ multiplication need not be commutative
  $\rightarrow$ multiplicative inverses need not exist
- special rings
  $\rightarrow$ semi-ring, near-ring, commutative ring, division ring, Lie ring

Lattices

- partially ordered sets, pairs of elements have unique supremum and infimum
- examples:
  $\rightarrow$ power set: supremum is union, infimum is intersection
  $\rightarrow$ natural numbers: supremum is LCM, infimum is GCD

More on Algebraic Structures (cont.)

Modules

- generalizes vector spaces with scalar fields
  $\rightarrow$ assume commutative/non-commutative rings instead of fields
- modules are additive abelian groups
  $\rightarrow$ multiplication is distributive over addition
- assumed in commutative algebras
  $\rightarrow$ closure and absorption properties of additions and multiplications (ideals)
- assumed in homological algebras
  $\rightarrow$ homological functors, chain complexes, category theory

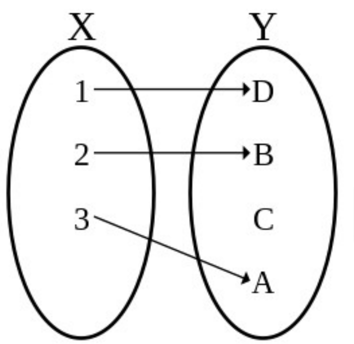
Algebras over a field

- general algebraic structure
- a vector space equipped with a bilinear product
  $\rightarrow$ field of elements with addition, (scalar) multiplication, and set of axioms

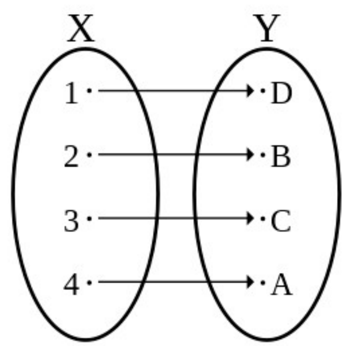
Algebra	vector space	bilinear operator	associativity	commutativity
complex numbers	R^2	product of complex numbers	Yes	Yes
cross product of 3D vectors	R^3	cross product	No	No
quaternions	R^4	Hamilton product	Yes	No
polynomials	R[X]	polynomial multiplication	Yes	Yes
square matrices	$R^{n\times n}$	matrix multiplication	Yes	No

Maps

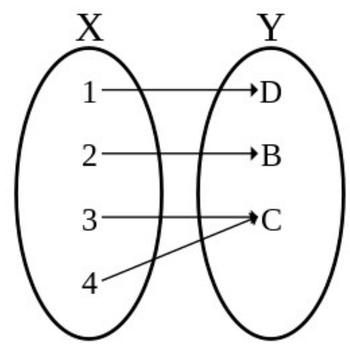
Injection:
injective &
non-surjective



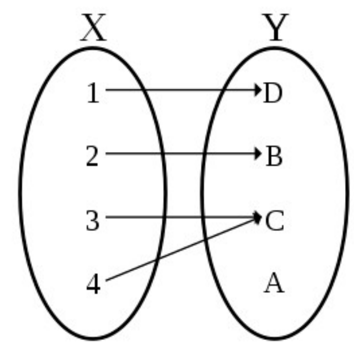
Bijection:
injective &
surjective



Surjection:
non-injective &
surjective



–
non-injective &
non-surjective



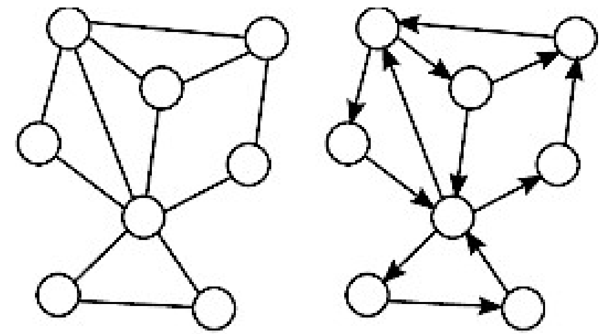
Inverse maps

right-inverse	$f_R^{-1}: \ Y \mapsto X$	$\Leftrightarrow$	$f(f_R^{-1}(y)) = y$ for $\forall y \in Y$
left-inverse	$f_L^{-1}: \ Y \mapsto X$	$\Leftrightarrow$	$f_L^{-1}(f(x)) = x$ for $\forall x \in X$
bijection	$f: \ X \mapsto Y$	$\Leftrightarrow$	$f(f_R^{-1}(y)) = f_L^{-1}(f(x))$
composition	$(f \circ g)^{-1}$	$\Leftrightarrow$	$g^{-1} \circ f^{-1}$

Maps as binary relations

- key properties: uniqueness, symmetry, composition

Graph Concepts



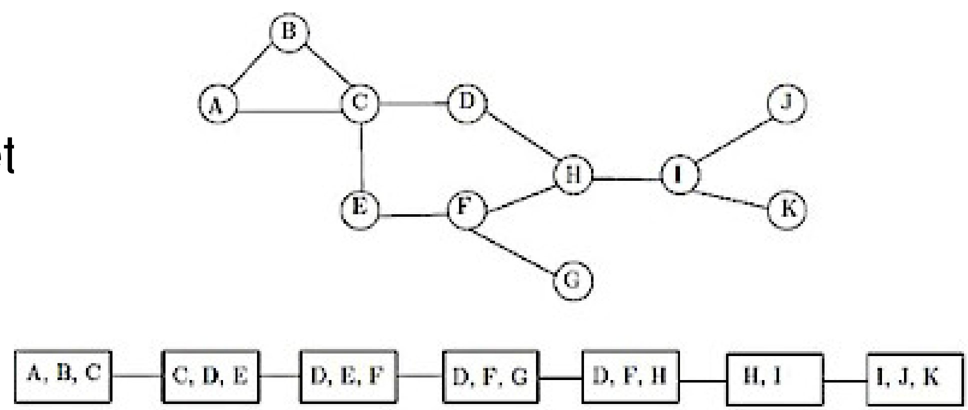
Graphs

- collection of edges E over vertices V
  → $|V|$ can be infinite
- edges can be generalized to k-simplexes

Graph (topological) embedding

- representation of G on a surface (manifold)
  → arcs representing edges cannot intersect
  → the surface is a union of regions (faces)
- any finite graph can be embedded in Euclidean space $\mathbb{R}^3$
  → a planar graph can be embedded in Euclidean space $\mathbb{R}^2$

Path decomposition

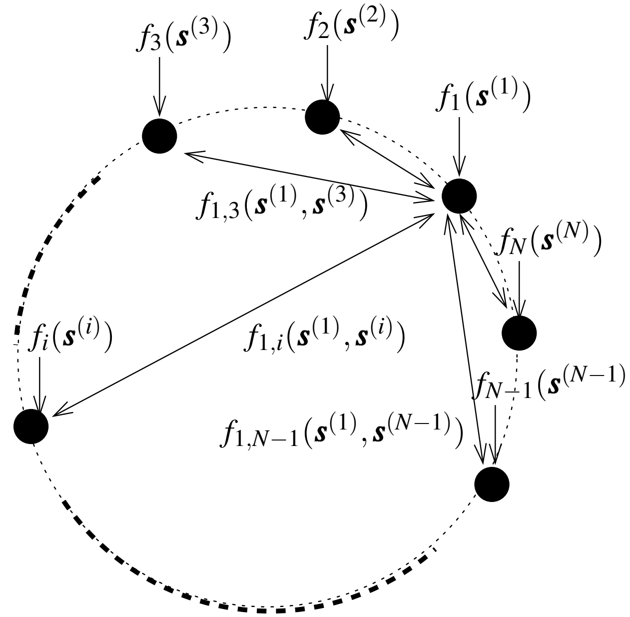


- a sequence of vertex subsets
  → all edges cross neighboring subset
- path-width
  → size of largest subset minus one
  → closely related to tree-width and tree decomposition

Graph Concepts (2)

Value assignment

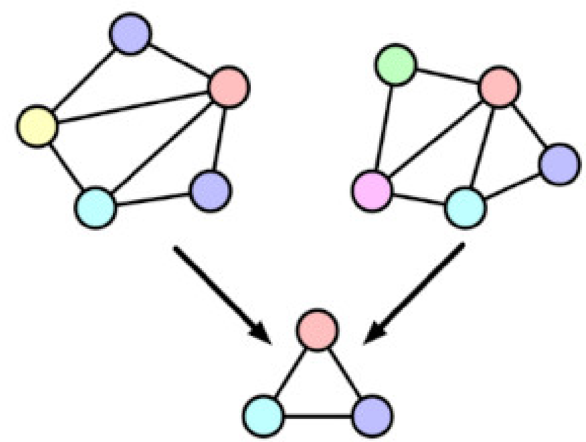
- permutation invariance of vertices



$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}_0 + \sum_{i=1}^{I} \boldsymbol{f}_i(\boldsymbol{x}_i) + \sum_{\substack{i,j=1\\ i\neq j}}^{I} \boldsymbol{f}_{i,j}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \cdots + \sum_{i=1}^{I} \boldsymbol{f}_{\{1:I\}\setminus i}(\boldsymbol{x})$$

Graph matching

- finding a similarity between (sub-) graphs
- exact matching: graph isomorphism
- inexact matching
  $\rightarrow$ best possible
- methods: pairings and optimizations



P. Loskot, "Analyzing Complex Models by Orthogonal Input-Output Decompositions," In Proc. Explainability, 2024.

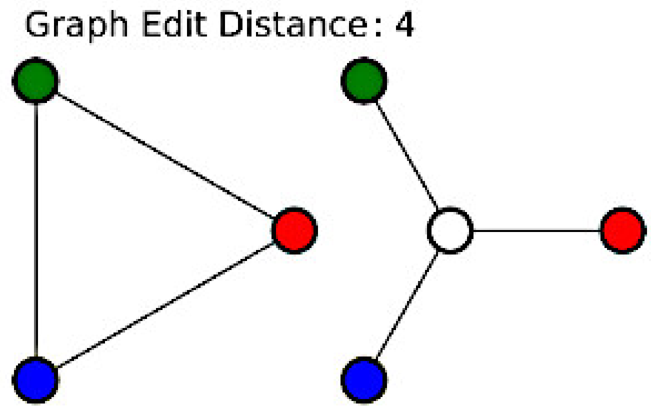
Graph Concepts (3)

Graph edit distance (GED)

- measure of similarity between two graphs
- smallest cost of changing one graph into another
  → insert/delete/substitute vertices and edges

GED example

1. remove edge between any 2 colors
2. change the 3rd color to white
3. add a vertex of now missing color
4. connect new vertex to white vertex



Graph isomorphism (GI)

- exact (sub-) graph matching

$$\text{bijection: } V(G_1) \mapsto V(G_2) \quad s.t. \quad \text{adjacency}$$

- structural similarity (isomorphism classes) of many other math objects
- key challenge
  → finding algorithms within given complexity classes
  (GI-complete and GI-hard problems)

Matroids

Definition

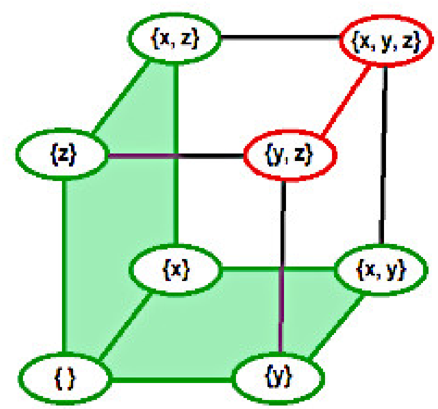
- matroid M consists of
  $\rightarrow$ finite ground set E and a collection of independent subsets $\mathcal{I}$ of E
- maximal $\mathcal{I}$ is a basis of the matroid
  $\rightarrow$ generalization of basis in linear algebra (span, rank, ...)
- a circuit is a minimal dependent subset of E
  $\rightarrow$ a cycle in graph representation of the matroid
- connection to linear algebra
  $\rightarrow$ E is subset of a vector space

Direct sum

- disjoint unions $E_1 \cup E_2$ and $\mathcal{I}_1 \cup \mathcal{I}_2$
- matroid M that cannot be written as $M_1 + M_2$ is connected or irreducible
- allows partitioning into a sum of matroids and maximum matching

Example matroid

- ground set $\{x, y, z\}$
- $\{y, z\}$ and $\{x, y, z\}$ dependent (in red)
- other subsets independent (green)



Arithmetic Graph Sums

Task

- given graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, define $G_1 + G_2$

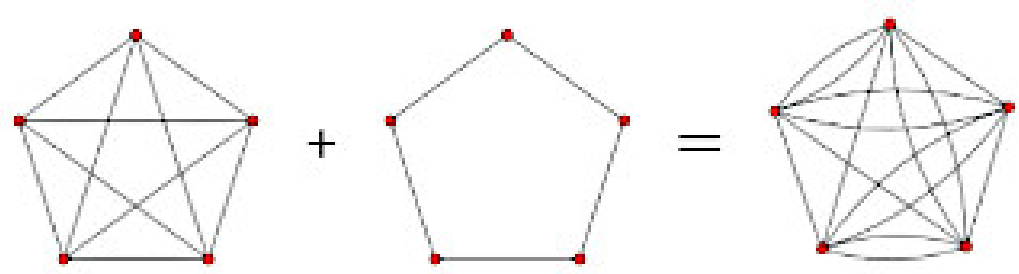
Solution

- shared vertices: $V_{1\cap 2} = V_1 \cap V_2$
- zero-pad adjacency matrices to the same size, $A_{1,2} \in \{0,1\}^{n\times n}$
- permute adjacency matrices to have the same ordering of shared vertices
- the sum is a multi-graph (integer additions)

$$[A_1]_{ij} + [A_2]_{ij} = [A_{1+2}]_{ij} \in \{0,1,2\}^{n\times n}$$

- the sum is a graph (or-additions)

$$[A_1]_{ij} \vee [A_2]_{ij} = [A_{1+2}]_{ij} \in \{0,1\}^{n\times n}$$



$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 1 & 1 & 2 \\ 2 & 0 & 2 & 1 & 1 \\ 1 & 2 & 0 & 2 & 1 \\ 1 & 1 & 2 & 0 & 2 \\ 2 & 1 & 1 & 2 & 0 \end{bmatrix}$$

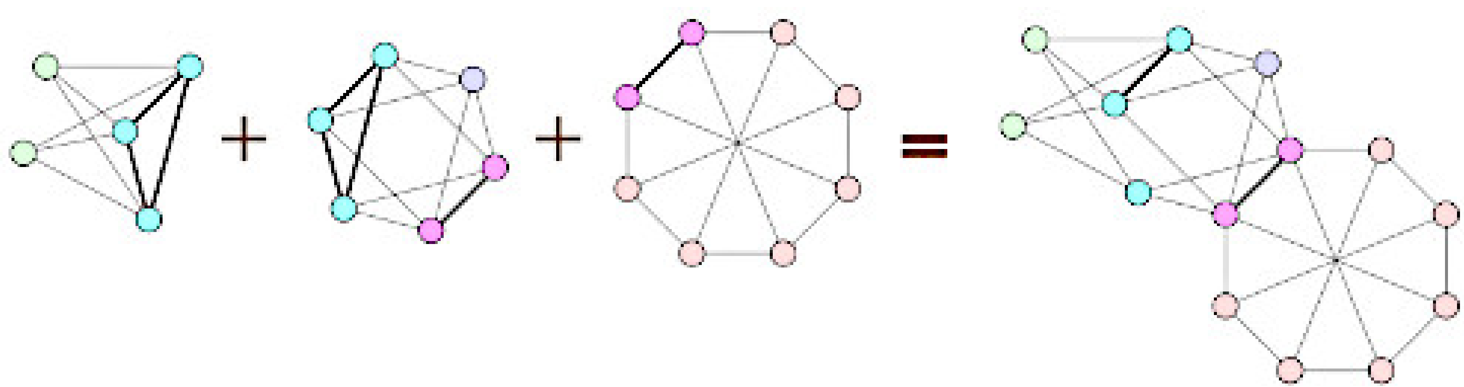
Clique Graph Sums

Task

- attach two or more graphs at their cliques of equal sizes
  → analogous to connected sums (set sums) of manifolds in topology

Solution

- reuse (share) the cliques
  → identify matching vertexes in these cliques
  → delete all (or some of) the clique edges
- k-clique sum
  → the shared clique to have (or at most) k vertices
- more than two graphs $((G_1+G_2)+G_3)+G_4\cdots$



- alternatively, a way of decomposing graphs into simpler graphs
  → usually constrained by closure under minor graph operations

Graph Products

Strategy

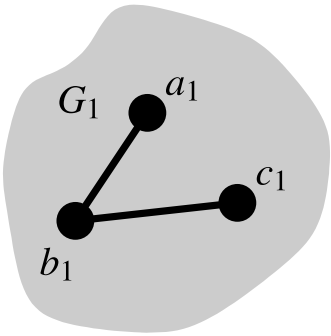
- multiply two graphs in a sense of combining vertexes
  $\rightarrow$ assume Cartesian product of two vertex sets
- then define rules to add edges (connecting combined vertices)
  $\rightarrow$ special rules for treating self-loops

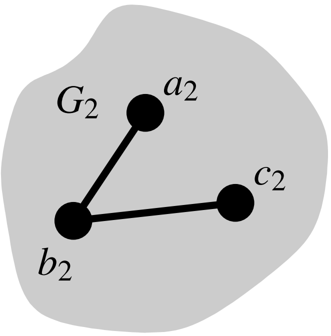
Notations

- $G_1 = G_2$: the graphs are naturally isomorphic
- product is a binary mapping:

$$G_1(V_1, E_1) \times G_2(V_2, E_2) \mapsto G_{12}(V_{12}, E_{12})$$

  $\rightarrow$ may or may not be commutative
- connectivity (graph edges)
  $\rightarrow$ edge: $(a_1 \sim a_1) \in E_1$, no edge: $(a_1 \nsim b_1) \notin E_1$
  $\rightarrow$ edge: $(a_2 \sim a_2) \in E_2$, no edge: $(a_2 \nsim b_2) \notin E_2$
- connectivity between combined vertices
  $\rightarrow (a_1, a_2) \sim (b_1, b_2)$, or, $(a_1, a_2) \nsim (b_1, b_2)$
- cardinalities: $v_{1,2} = |V_{1,2}|$ and $e_{1,2} = |E_{1,2}|$

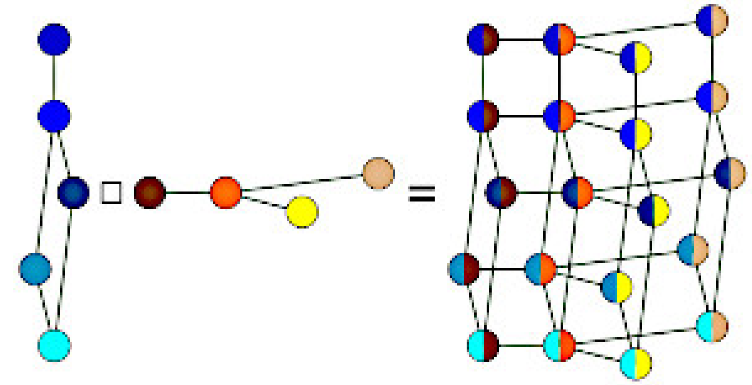




Cartesian Graph Product

Definition of $G_1 \square G_2$

- vertices are Cartesian product $V_1 \times V_2$
- edges $(a_1, a_2) \sim (b_1, b_2)$ if and only if:
  $a_1 = b_1$ and $a_2 \sim b_2$, or, $a_1 \sim b_1$ and $a_2 = b_2$
  $\rightarrow |(a_1, a_2) \sim (b_1, b_2)| = v_1 e_2 + e_1 v_2$
- also known as box product
  (the box indicates that four edges are from Cartesian product of two edges)



Properties

- relationship of adjacency matrices

$$\boldsymbol{A}_{1\square 2} = \boldsymbol{A}_1 \otimes \boldsymbol{I}_{n_2} + \boldsymbol{I}_{n_1} \otimes \boldsymbol{A}_2 \in \{0,1\}^{(n_1 n_2)\times(n_1 n_2)}$$

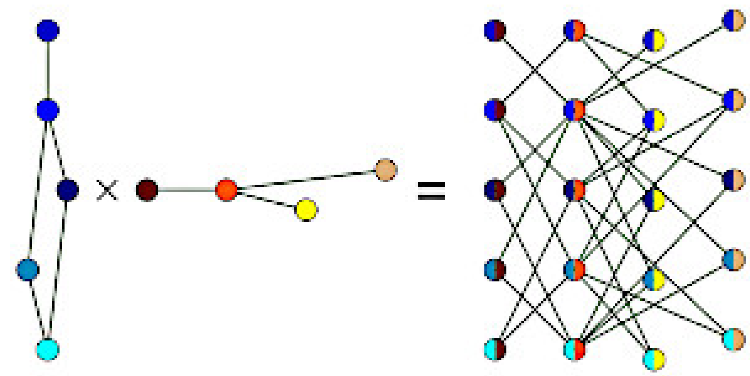
- commutative: $G_1 \square G_2 = G_2 \square G_1$
  $\rightarrow$ may not hold for labeled graphs
- associative: $(G_1 \square G_2) \square G_3 = G_1 \square (G_2 \square G_3)$

Problems

- constructing regular graphs as Cartesian products
- Cartesian factorization of graphs (finding prime factors)
  $\rightarrow$ e.g. Cartesian product is bipartite *iff* each factor is bipartite

Tensor Graph Product

Definition of $G_1 \times G_2$



- vertices are Cartesian product $V_1 \times V_2$
- edges $(a_1, a_2) \sim (b_1, b_2)$ if and only if:
  $a_1 \sim b_1$ and $a_2 \sim b_2$
  $\rightarrow |(a_1, a_2) \sim (b_1, b_2)| = 2e_1e_2$
- known as direct, Kronecker, categorical, cardinal, relational product, or conjunction

Properties

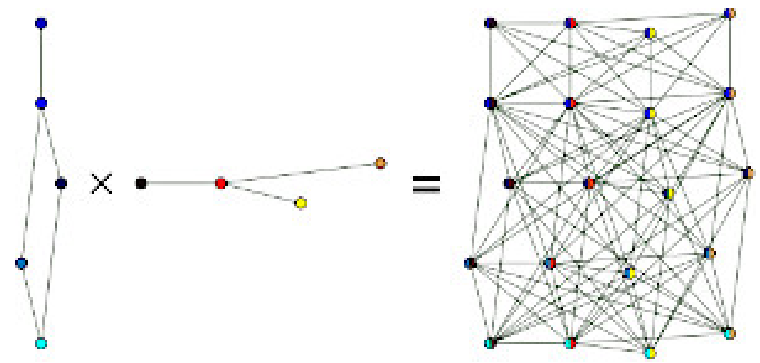
- relationship of adjacency matrices

$$\boldsymbol{A}_{1\times 2} = \boldsymbol{A}_1 \otimes \boldsymbol{A}_2$$

  $\rightarrow$ Kronecker (tensor) product
- tensor products do not have unique factorization, but they do have the same number of irreducible factors
  $\rightarrow$ e.g. if either G_1 or G_2 is bipartite, then so is their tensor product
- tensor product is a category-theoretic product in the category of graphs and graph homomorphisms
  $\rightarrow$ imposes certain structures of categories

Lexicographic Graph Product

Definition of $G_1 \cdot G_2$



- vertices are Cartesian product $V_1 \times V_2$
- edges $(a_1, a_2) \sim (b_1, b_2)$ if and only if:
  $a_1 \sim b_1$ or ($a_1 = b_1$ and $a_2 \sim b_2$)
  $\rightarrow |(a_1, a_2) \sim (b_1, b_2)| = v_1 e_2 + e_1 v_2^2$

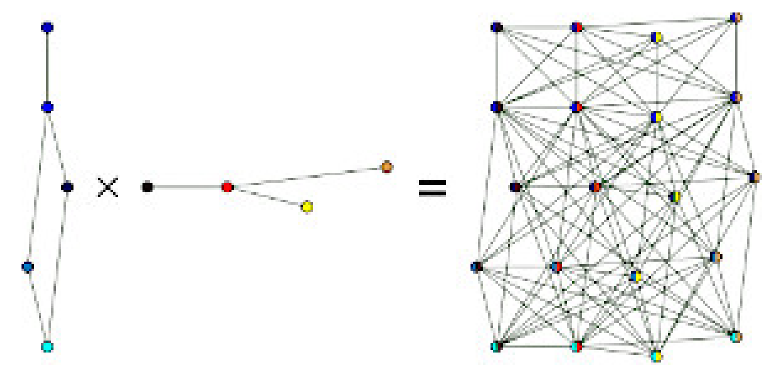
Properties

- if edges represent order relations, then edges of $G_1 \cdot G_2$ represent the corresponding lexicographic order
- general non-commutative: $G_1 \cdot G_2 \neq G_2 \cdot G_1$
- distributive law: $(G_1 + G_2) \cdot G_3 = G_1 \cdot G_2 + G_2 \cdot G_3$ ('+' represents union)
- complements: $\overline{G_1 \cdot G_2} = \overline{G}_1 \cdot \overline{G}_2$
- identities exist for:
  $\rightarrow$ independence number
  $\rightarrow$ clique number
  $\rightarrow$ chromatic number
  …

Strong Graph Product

Definition of $G_1 \boxtimes G_2$

- vertices are Cartesian product $V_1 \times V_2$
- edges $(a_1, a_2) \sim (b_1, b_2)$ if and only if:
  ($a_1 = b_1$ and $a_2 \sim b_2$) or
  ($a_1 \sim b_1$ and $a_2 = b_2$) or
  ($a_1 \sim b_1$ and $a_2 \sim b_2$)
  $\rightarrow |(a_1, a_2) \sim (b_1, b_2)| = v_1 e_2 + e_1 v_2 + 2 e_1 e_2$
- union of Cartesian and tensor products



Properties

- clique number of strong product equals the product of their clique numbers
- some (many) properties shown only for specific instances of graphs
  $\rightarrow$ and specific (specialized) attributes

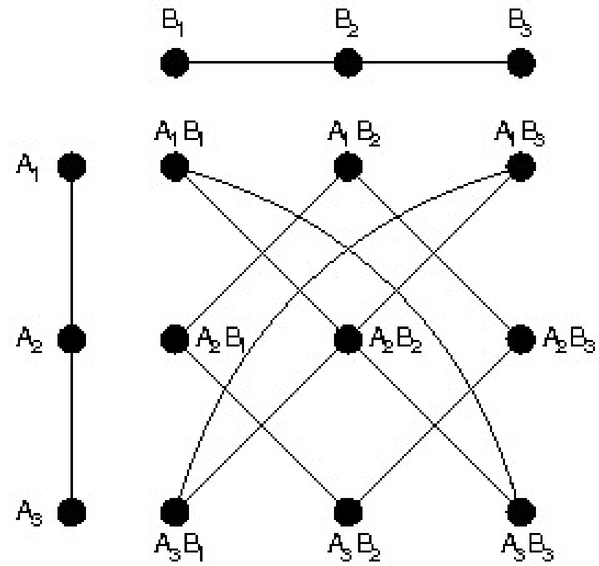
Problems

- decompositions of planar graphs into strong products
  $\rightarrow$ can be used to show many properties of planar graphs

Other Graph Products

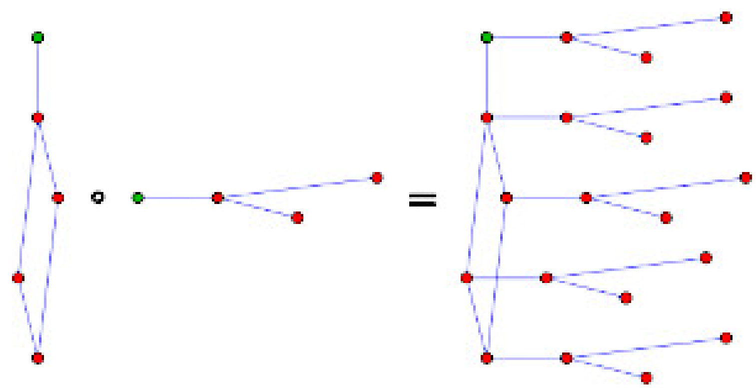
Modular graph product

- vertices are Cartesian product $V_1 \times V_2$
- edges $(a_1, a_2) \sim (b_1, b_2)$ if and only if:
  $(a_1 \sim b_1$ and $a_2 \sim b_2)$ or
  $(a_1 \nsim b_1$ and $a_2 \nsim b_2)$
- often used to reduce subgraph isomorphism into problem of finding graph cliques



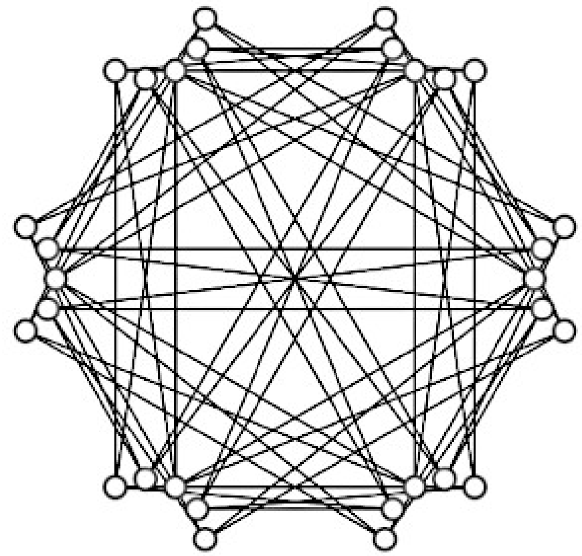
Rooted graph product

- take $|V_1|$ copies of G_2, and identify all $a_1 \in G_1$ with the root vertex of i-th copy of G_2
- rooted product of $G_1 = G_2$ is a subgraph of their Cartesian product
- mainly assumed for trees
  $\rightarrow$ rooted product of two trees is another tree



Other Graph Products (cont.)

Zig-zag graph product



- defined for regular graphs, denoted as $G_1 \circ G_2$
- replaces each vertex of G_1 with a copy (cloud) of G_2
  $\rightarrow$ then zig-zag interconnects these clouds
- approximately inherits size of larger G_1 and degree of smaller G_2
- if G_2 is expander graph, then the expansion is only slightly worse than expansion of G_1

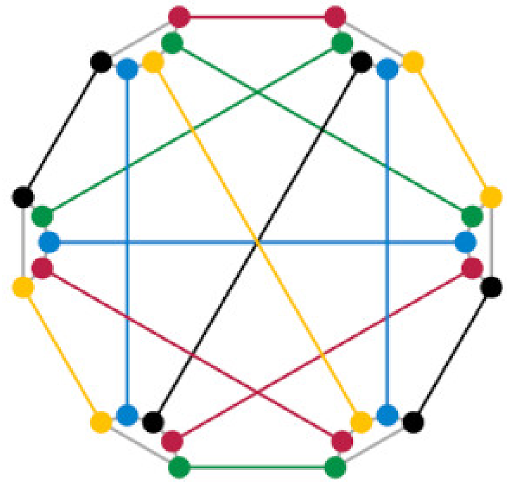
Expander graphs

- a finite, undirected multi-graph with a few constraints
- used in construction of sparse graphs to expand their vertices, edges or other properties (e.g. spectral expanders)
- every connected graph is expander, but with different expansion parameters
  $\rightarrow$ good expander has low degree and yields high expansion
- used extensively to construct graphs (networks) with desired properties

Other Graph Products (cont.)

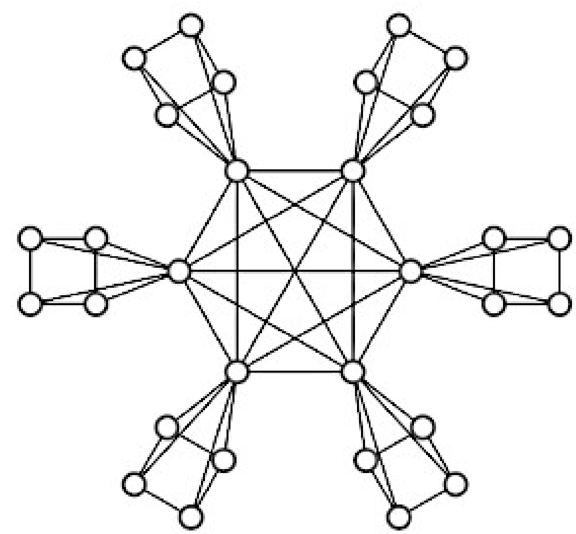
Replacement graph product $G_1 \circledR G_2$

- incorporates another graph product
- the aim is reduce the degree of a graph while maintaining its connectivity
- regular graphs with neighbors ordering
- vertices are Cartesian product $V_1 \times V_2$
- edge rules a bit more complicated



Corona graph product $G_1 \circ G_2$

- incorporates another graph product
- choose G_1 as the center graph
- take $|V_1|$ copies of G_2
- each vertex in G_1 is attached to each vertex in one copy of G_2



Summary of Graph Products

Cartesian product (box product) $G_1 \square G_2$	$a_1 = b_1 \wedge a_2 \sim b_2$ $\vee$ $a_1 \sim b_1 \wedge a_2 = b_2$
Tensor product (Kronecker product, categorical product) $G_1 \times G_2$	$a_1 \sim b_1 \wedge a_2 \sim b_2$
Lexicographical product $G_1 \cdot G_2$ or $G_1[G_2]$	$a_1 \sim b_1$ $\vee$ $a_1 = b_1 \wedge a_2 \sim b_2$
Co-normal product (disjunctive product, OR product) $G_1 * G_2$	$a_1 \sim b_1$ $\vee$ $a_2 \sim b_2$
Modular product	$a_1 \sim b_1 \wedge a_2 \sim b_2$ $\vee$ $a_1 \not\sim b_1 \wedge a_2 \not\sim b_2$
Homomorphic product[1][3] $G_1 \ltimes G_2$	$a_1 = b_1$ $\vee$ $a_1 \sim b_1 \wedge a_2 \not\sim b_2$
Strong product (Normal product, AND product) $G_1 \boxtimes G_2$	$a_1 = b_1 \wedge a_2 \sim b_2$ $\vee$ $a_1 \sim b_1 \wedge a_2 = b_2$ $\vee$ $a_1 \sim b_1 \wedge a_2 \sim b_2$

https://en.wikipedia.org/wiki/Graph_product

Other Graph Operations

Task

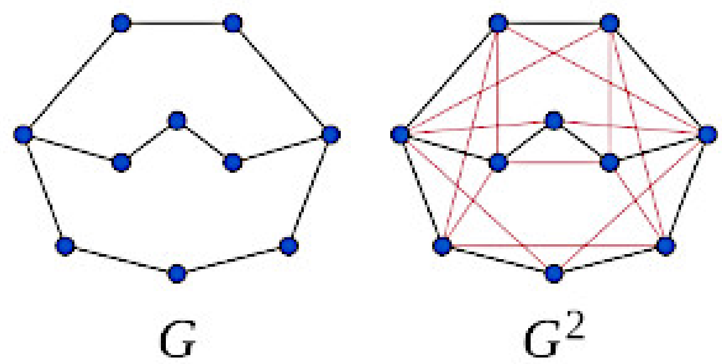
- transform input graphs into output a graph
  $\rightarrow$ define unary (transforms) and binary operators
- directed and undirected graphs, weighted and unweighted graphs
- to be used in graph calculus, algebras, generative grammars etc.

Elementary operations

- simple editing of vertices and edges
  $\rightarrow$ e.g. add, delete, merge, split (cf. GED)

Graph powers

- if G has adjacency matrix $\boldsymbol{A}$, then G^k has adjacency matrix $\boldsymbol{A}^k$
- different from a sequence of products $G \times G \times \cdots$



Unary graph transformations

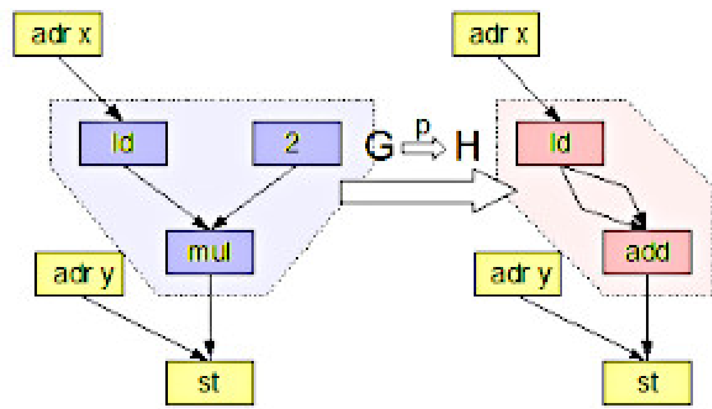
- reverse, minor, dual, complement, medial, quotient, Mycielskian

OTHER GRAPH OPERATIONS (CONT.)

Graph rewriting

- rules for transforming a host graph G_1 into a replacement graph G_2
  → often defined algorithmically
- can be restricted to a matched subgraph pattern
  → pattern matching requires solving the subgraph isomorphism problem
- can be refined by exploiting labeling

Applications of graph rewriting



- describe non-stationary dynamic system
  → set of system states varies in time
- graph language with grammars
  → enumeration of all graphs
- graph model of a system
  → graph optimization

Abstract rewriting systems

- generalization of graph rewriting
- rules to transforming (reducing) sets of objects and their relations
  → also allow rewriting the rules themselves

Other Graph Operations (cont.)

Binary graph operations

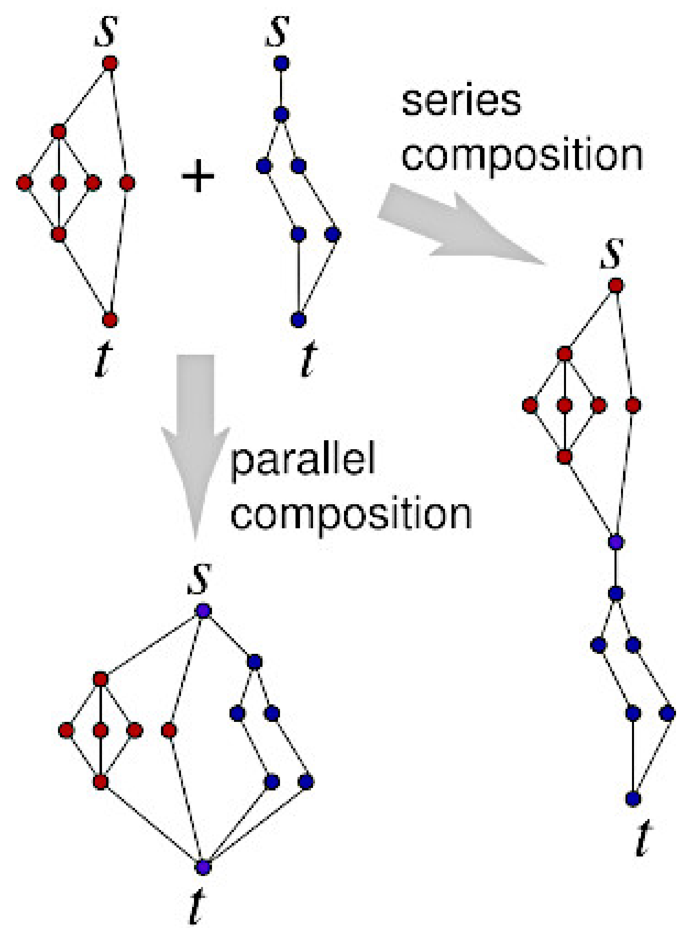
- (disjoint) union $G_1 \cup G_2$ and intersection $G_1 \cap G_2$
- joints $G_1 \nabla G_2$
- graph sums and products

Graph composition

- series-parallel graphs with two terminal vertices
- formed recursively by two simple operations
  → parallel composition
  → series composition
- other such operations can be defined
  → (non-) commutative operations
  → generalizations of recursive constructions

Application of graph SP composition

- designing and analyzing circuits
  → network flows



Algebraic Graph Theory

Spectral theory of graphs

- using linear algebra to study properties of matrix representations of graphs
  $\rightarrow$ adjacency and Laplacian matrices
  $\rightarrow$ eigenvalues and eigenvectors defined by the characteristic polynomial
- the eigenvalues represent the graph spectrum
  $\rightarrow$ a multiset of eigenvalues

Key properties

- the graph spectrum is invariant of vertex ordering
- the graphs with the same spectrum are isospectral
  $\rightarrow$ isomorphic graphs have the same spectrum
  $\rightarrow$ isospectral graphs does not have to be isomorphic

Applications of graph spectrum

- graph signal processing
  $\rightarrow$ eigen-decomposition is a DFT
- problems in topology and geometry
- studying and designing networks
  $\rightarrow$ flows and capacity, connectivity, synchronization, ...

Algebraic Graph Theory (cont.)

Group theory of graphs

- various forms of symmetry, transitivity, and regularity
- can be related to graph spectra

Graph invariants

- the invariants can be:
  → true/false indicators
  → numbers: positive integers, real values
  → sequence (vector) of numbers
  → polynomials
- can be additive or multiplicative
  → assuming disjoint union of graphs G_1 and G_2

$$\mathcal{P}(G_1) + \mathcal{P}(G_2) = \mathcal{P}(G_1 \cup G_2)$$

$$\mathcal{P}(G_1) \times \mathcal{P}(G_2) = \mathcal{P}(G_1 \cup G_2)$$

- if $G_1 = G_2$ (isomorphic), then $\mathcal{P}(G_1) = \mathcal{P}(G_2)$
  → opposite statement may not be true
  → opposite statement can be used to identify $G_1 \neq G_2$ (non-isomorphism)

Graph Signal Processing

Auto-regressive modeling

- 1D signal filtering

$$s_{t-1} = z^{-1} s_t, \qquad h(z) = h_0 z^0 + h_1 z^{-1} + \ldots + h_{N-1} z^{-(N-1)}$$

- graph filtering

$$\boldsymbol{s}_{t-1} = \boldsymbol{A}^{-1} \boldsymbol{s}_t, \qquad h(\boldsymbol{A}) = h_0 \boldsymbol{A}^0 + h_1 \boldsymbol{A}^{-1} + \ldots + h_{N-1} \boldsymbol{A}^{-(N-1)}$$

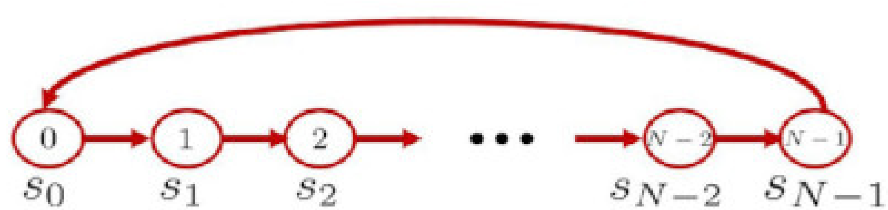
- shift-invariance

$$h'(z) \to z\, h(z), \qquad h'(\boldsymbol{A}) \to \boldsymbol{A}\, h(\boldsymbol{A})$$

Graph Fourier transform

$$\boldsymbol{s}_{\text{out}} = \boldsymbol{A}\, \boldsymbol{s}_{\text{in}}, \quad \boldsymbol{A} = \boldsymbol{U}^{-1} \boldsymbol{\Lambda} \underbrace{\boldsymbol{U}}_{\text{GFT}}$$

$$\boldsymbol{U}\, \boldsymbol{s}_{\text{out}} = h(\boldsymbol{\Lambda})\, \boldsymbol{U}\, \boldsymbol{s}_{\text{in}}$$



P. Loskot, Tutorials on Graph Signal Processing, SIGNAL 2021 and 2022.

Graph Signal Processing (cont.)

Generalization

1. signals themselves are graphs
2. graphs can be rewritten between time-steps

Simple case

- given $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{S}_0$

$$\boldsymbol{S}_{t+1} = \boldsymbol{A}\boldsymbol{S}_t + \boldsymbol{B}, \quad t = 0, 1, 2, \ldots \quad \Rightarrow \quad \boldsymbol{S}_t = \boldsymbol{A}^t \left(\boldsymbol{S}_0 - \frac{\boldsymbol{B}}{\boldsymbol{I} - \boldsymbol{A}} \right) + \frac{\boldsymbol{B}}{\boldsymbol{I} - \boldsymbol{A}}$$

With rewriting

- account for adding/removing/reordering vertices

$$\boldsymbol{S}_{t+1} = \mathcal{R}[\boldsymbol{A}] \cdot \mathcal{R}[\boldsymbol{S}_t] + \mathcal{R}[\boldsymbol{B}], \quad t = 0, 1, 2, \ldots$$

- consequently
  $\rightarrow$ $\boldsymbol{A}$ and $\boldsymbol{B}$ must also be rewritten
- becomes graph filtering problem
  $\rightarrow$ structure and numerical values over that structure
- the key is to define $\mathcal{R}[\cdot]$ rewriting operators

Take-Home Messages

Present numerical modeling techniques

- numerical values arranged as vectors and matrices
  → locally in case of manifolds
- models of dynamic systems
  → auto-regressive modeling and calculus

A new generation of semantic modeling

- define models involving arithmetic operations over mathematical objects
  → sums and products as basic binary operations
  → various unary transformations

This talk

- considered graphs as mathematical objects
- outlined different graph sums and products
- discussed graph rewriting as a class of unary transformations
- mentioned autoregressive graph filtering

Future applications

- semantic models may be naturally interpretable (machine learning)
- bring key results and ideas from applied/basic math into engineering
  → geometry, topology, algebras, category theory etc.

Thank you!

pavelloskot@intl.zju.edu.cn

Recommended Resources

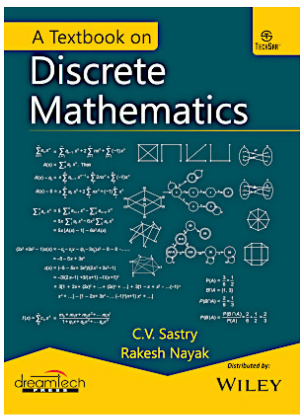
Graph operations (quick overviews and key results)

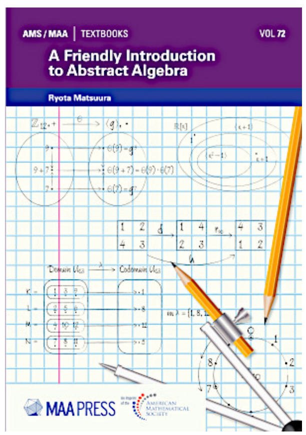
https://en.wikipedia.org/wiki/Graph_operations

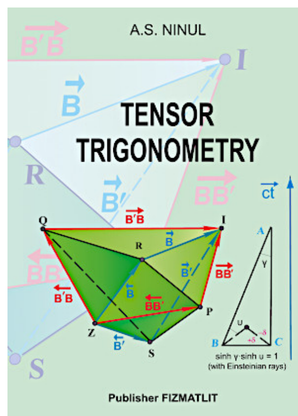
https://en.wikipedia.org/wiki/Graph_product

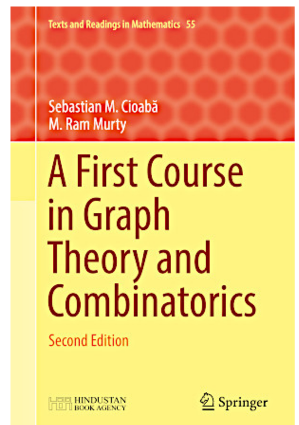
https://en.wikipedia.org/wiki/Graph_rewriting

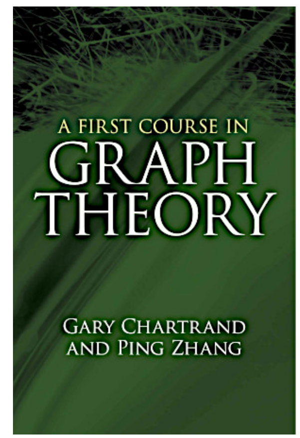
Books (there are many other such books)

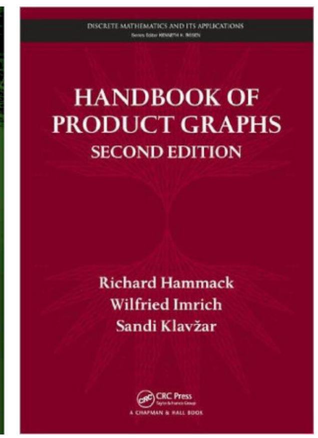












Algebraic Concepts (selected applied math topics for SP/ML)

https://www.iaria.org/conferences2023/filesSIGNAL23/PavelLoskot_Keynote_AlgebraicConcepts.pdf

https://www.iaria.org/conferences2024/filesDigitalSustainability24/PavelLoskot_Keynote_IntroductionToCurves.pdf