MULTILEVEL GRAPHS: A DYNAMIC MATRIX APPROACH

Edward Muñoz Garro
Department of Computer Science and Informatics
University of Costa Rica, San José, Costa Rica
0000-0002-8016-3787

ABSTRACT:

This paper presents a dynamic matrix framework for the normalization and simplification of complex systems using second-order multilevel graphs. Building on the visual alternative proposed by Muñoz Garro [1], we represent a system as a block adjacency matrix whose entries are integrable functions of edge weights. We compute the derivative of this matrix with respect to a continuous parameter to quantify the rate of structural change and integrate it to capture the cumulative influence of each connection. Our mixed methodology combines a theoretical formalization of the correspondence between subgraphs and submatrices—with conditions for continuity and integrability—and an experimental phase of Python simulations on three case studies: industrial process networks, corporate information flows, and synthetic graphs. The results show that the structural derivative identifies critical nodes with high agreement with established metrics, that a small fraction of edges accounts for most of the system complexity, and that the method achieves substantial reduction in structural complexity while preserving essential connectivity. Computational costs grow cubically, indicating potential scalability considerations. We conclude that this model offers a robust quantitative complement to visual techniques and supports automation and extension to higher-order graphs.

Keywords: Multilevel graphs; block adjacency matrix; dynamic matrix derivative; structural complexity; complex systems normalization.

I. INTRODUCTION

The normalization of complex systems plays a crucial role in areas such as software engineering, database design, and process optimization, where structural clarity and dependency control determine both maintainability and performance [2], [3]. Muñoz Garro [1] introduced a visual alternative using second-order graphs, combining Armstrong's axioms with second-order logic to break cycles and detect patterns without compromising the integrity of functional dependencies. Although this technique provides strong intuition, it lacks a quantitative framework to dynamically measure change rates and the cumulative influence of connections in multilevel networks.

To address this limitation, we model a complex system as a second-order multilevel graph and construct its block adjacency matrix A(t), whose entries are integrable functions of edge weights W_ij (t). The derivative dA/dt identifies critical normalization points in real time [10], while the integral ∫A(t) dt quantifies accumulated influence over intervals of interest.

This study, conducted at the University of Costa Rica from March 2023 to April 2025 without external funding, follows a mixed approach: in the theoretical phase, we formalize the correspondence between subgraphs and submatrices and define continuity, and integrability conditions, and in the experimental phase, we implement Python simulations on three representative case studies. The paper is structured as follows: Section II reviews the theoretical framework of multilevel graphs and dynamic matrix calculus; Section III describes the methodology; Section IV presents quantitative results; Section V discusses implications, limitations and optimizations; and Section VI concludes with contributions and future directions, including extension to higher-order graphs.

A. Background

Normalization of complex systems, notably relational database schemas, is grounded in Armstrong's axioms, which define inference rules for functional dependencies and guarantee normal-form correctness [6]. Manual application becomes cumbersome as attributes and dependencies scale. Graph theory offers a visual framework for modeling elements as vertices and dependencies as edges [5]. While formal algebraic algorithms optimize

ISSN: 2015-5392

dependency sets, they often lack intuitive insight. Attallah [6] introduced an interactive 3D simulation for schema decomposition, and Di Battista et al. [8] [MC1.1][MC1.2][EMG1.3]proposed planar-graph schema representations; however, neither provides automated cycle removal. Harel's Statecharts [9] introduced nested and concurrent graph structures, inspiring multilevel grouping. Computational methods such as GraphSteal [10] and Joint Graph Layouts [11] demonstrate dynamic partitioning and mesh visualization to enhance performance and clarity. Shi et al.'s Top-K graph pattern matching [12] applies second-order logic to detect subgraphs under global constraints.

B. Justification

Although Armstrong's axioms [4] furnish a solid mathematical basis for normalization, their algebraic nature can be inaccessible. Visual approaches [6], [8] improve comprehension but omit automated mechanisms for eliminating cyclic or redundant dependencies. Hierarchical models such as Statecharts [9] support compound nodes without addressing dependency simplification. Optimization and pattern-detection techniques [10], [14] yield performance benefits yet fail to integrate multilevel visualization with logical validation. This fragmentation reveals the absence of a unified framework. Our work fills this gap by introducing second-order graphs as a combined visual and axiomatic tool, enabling multilevel grouping, systematic cycle breaking, and second-order logical validation within a single normalization process.

II. THEORETICAL FRAMEWORK

A. Armstrong's Axioms and Functional Dependencies

Armstrong's axioms—reflexivity (if $Y \subseteq X$ then $X \to Y$), augmentation (if $X \to Y$ then $XZ \to YZ$), and transitivity (if $X \to Y$ and $Y \to Z$ then $X \to Z$)—form a sound and complete inference system for deriving all functional dependencies in a relational schema. Given an initial set F, these rules yield its closure F^+, ensuring that normalization preserves every necessary constraint without redundancy [3].

B. Second-Order Logic

Second-order logic extends first-order logic by allowing quantification over predicate and set variables rather than only individuals. This enhanced expressivity is crucial for specifying multiscale properties in second-order graphs—e.g., constraints on entire subgraph collections—that cannot be captured in first-order frameworks [12].

C. Second-Order Graphs and Hierarchical Modeling

A second-order (multilevel) graph $G = (V,E,\psi)$ permits each vertex $v_i \in V$ (and each edge) to encapsulate a subgraph $\psi(v_i)$, enabling recursive hierarchies of dependencies. Such a representation simplifies the detection of cycles and patterns across nested levels while preserving conceptual clarity [1].

D. Block Adjacency Matrix

For a multilevel graph comprising k principal subgraphs of sizes $n_1,...,n_k$, the global adjacency matrix A partitions into k×k blocks:

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1k} \\ A_{21} & A_{22} & \cdots & A_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k1} & A_{k2} & \cdots & A_{kk} \end{pmatrix}$$

Where each diagonal block A_{ii} encodes intra-graph connectivity, whereas each off-diagonal block A_{ij} $(i \neq j)$ captures inter-subgraph links [5].

E.Dynamic Matrix Calculus

Modeling A as a time-continuous function A(t) enables quantitative analysis of structural evolution. Its derivative

$$A'(t) = \frac{d}{dt} A(t)$$

Indicates instantaneous strengthening (positive entries) or weakening (negative entries) of dependencies. The accumulated influence over $[t_0, t_1]$ is given by

$$I = \int_{t_0}^{t_1} A(t)\,dt,$$

Where each I_{ij} measures the total weighted influence or duration of relation i→j [11].

F. Advantages Over Static Visual Methods

Static visual approaches—such as Muñoz Garro's second-order visualization—facilitate intuitive learning but do not scale efficiently to large systems or support real-time monitoring. The dynamic matrix framework complements visualization with algorithmic and quantitative metrics, enabling large-scale application, precision retention, and live tracking of dependency evolution [1], [6], [7].

ISSN: 2015-5392

III. METHODOLOGY

A. Approach

This study employs a quantitative framework based on dynamic matrix representations of second-order graphs. Three case studies—a synthetic industrial network, a hierarchical corporate-workflow graph, and an Erdős–Rényi random graph—are parameterized by continuous edge-weight functions $w_{nk}(t)$. From these we generate time series of block adjacency matrices $A(t_k)$, numerically compute the structural derivative $\dot{C}A(t_k)=dA/dt$ via finite differences and evaluate the accumulated influence $I=\int A(t)\,dt$. Results are validated against classical centrality measures, and established normalization criteria.

B. Units of Analysis

- Graph elements: Simple and their connecting edges, as defined by each case study.
- Matrix entries $A_{ij}(t)$: Instantaneous weight of relation $i \to j$.
- Temporal subseries $A_{ij}(t)$ and $\int A_{ij}(t)\,dt$: Used to assess dynamic and cumulative behavior per edge.

C. Data Collection

In the data-collection stage, we first generated three distinct graph topologies: a synthetic industrial network composed of chained block modules to mimic sequential process dependencies; a hierarchical corporate-workflow graph in which departments and their associated tasks form nested subgraphs; and an Erdős–Rényi random graph serving as an unstructured baseline.

For each topology, we then assigned time-varying edge weights $w_{ij}(t)$—using linear, exponential, or stochastic functions—to capture dynamic relationship strengths, sampling these functions uniformly over the interval $[t_0, t_1]$ at fixed time steps Δt to produce the series of adjacency matrices used in the subsequent analyses.

D. Analysis Procedures

- Construct $\{A(t_k)\}$ for $k=0...N$.
- Compute structural derivative by forward differences:
$$\dot{A}(t_k) \approx \frac{(A(t_{k+1}) - A(t_k))}{\Delta t}.$$
- Approximate the integral via the trapezoidal rule:
$$\int_{t_1}^{t_1} A(t)dt \approx \sum_{k=0}^{N-1} \frac{A(t_k) + A(t_{k+1})}{2} \Delta t.$$
- Extract performance metrics, including top-k node correspondence, edge-concentration ratios, and post-normalization complexity reduction.
- Perform scalability assessment: Record simulation runtimes as a function of graph size.

IV. RESULTS

In this section, we present the procedure for constructing the adjacency matrix of a second-order multilevel graph, defining its dynamic modeling through continuous functions and computing its structural derivative, as well as classifying fundamental relationships and analyzing their behavior both in the static matrix A and in its derivative $\dot{A}$. This methodological framework provides the quantitative foundation necessary to assess the evolution and simplification of complex systems, systematically integrating the visual approach proposed by Muñoz Garro [9] into a formal quantitative framework.

A. Static Adjacency Matrix

In this subsection, we rigorously develop the adjacency-matrix representation of a directed second-order multilevel graph. Let $G = (V, \psi_G)$, $|V| = n$ where ψ_G encodes all functional-dependency relations among vertices. The static adjacency matrix $A = [M_{ij}] \in \{0,1\}^{n\times n}$ is defined entrywise by

$$M_{ij} = \begin{cases} 1, & \text{if there is a directed edge } i \to j, \\ 0, & \text{otherwise.} \end{cases}$$

Each "1" therefore represents a dependency from i to j. When G decomposes into k principal subgraphs of sizes $n_1, \dots, n_k$, we reorder vertices to reveal a k×k block structure:

$$A = \begin{pmatrix} A_{11} & \cdots & A_{1k} \\ \vdots & \ddots & \vdots \\ A_{k1} & \cdots & A_{kk} \end{pmatrix},$$

Where each diagonal block $A_{ii} \in \{0,1\}^{n_i\times n_i}$ captures intra-subgraph connectivity, whereas each off-diagonal block $A_{ij} \in \{0,1\}^{n_i\times n_j}$ encodes inter-subgraph links.

Example: Let $V_1 = \{x, y, z\}$ and $V_2 = \{w, u\}$, with one internal edge $x \to y$ and one external edge $z \to w$. Ordering vertices (x, y, z, w, u) yields

$$A_{11} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, A_{12} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix}, A_{21} = 0_{2x3}, A_{22} = 0_{2x2},$$

ISSN: 2015-5392

so that

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

B. Dynamic Modeling and Structural Derivative

This block-matrix construction serves as the basis for dynamic modeling. By treating A as a continuous function A(t), we can compute its structural derivative

$$\dot{A} = \frac{dA(t)}{dt},$$

Which captures the instantaneous strengthening (positive entries) or weakening (negative entries) of dependencies. Furthermore, integrating A(t) over any interval quantifies cumulative influence:

$$\int_{t_0}^{t_1} A(t)dt,$$

Where each I_{ij} measures the total or weighted duration of relation i→j. In the following sections, we use this formalism both to classify fundamental relations and to compare behavior in the static matrix A versus its derivative $\dot{A}$, thereby providing a quantitative foudation for multilevel normalization and simplification.

C. Classification of Relations and Their Matrix Behavior

Point-to-Point

In a directed second-order graph, a point-to-point relation connects two simple vertices ααα and β in a one-to-one mapping:

$\psi = \{\alpha \to \beta\}$ [9]

Its adjacency matrix, when ordering the vertices as $(\alpha \to \beta)$, is

$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ [5].

If the weight of the edge α→ β varies continuously with parameter t as w(t), the dynamic adjacency matrix and its structural derivative become $A(t) = \begin{pmatrix} 0 & w(t) \\ 0 & 0 \end{pmatrix}, \dot{A}(t) = \frac{dA(t)}{dt} = \begin{pmatrix} 0 & \dot{w}(t) \\ 0 & 0 \end{pmatrix}$ [10].

Given the elementary nature of this relation—and its foundational role in our taxonomy—we do not explore it further and instead proceed to relations of higher complexity.

Dynamic Relation (One-to-Many)

In a dynamic (one-to-many) relation, a composite vertex C connects to a set of simple vertices $\{s_1, \dots, s_m\}$. Formally, $\psi_G \supseteq \{C \to s_i : i = 1, \dots, m\}$.

When ordering the vertices as $(C, s_1, \dots, s_m)$, the adjacency matrix for this subgraph takes the block form

$$A = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix},$$

Where the first row has ones under each s_i and zeros elsewhere.

If each edge $C \to s_i$ carries a continuous weight $w(t)$, its structural derivative becomes

$$\dot{A}(t) = \begin{pmatrix} 0 & \dot{w}_1(t) & \dot{w}_2(t) & \cdots & \dot{w}_m(t) \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix},$$

Where $\dot{w}_i(t) > 0$ indicates strengthening of $C \to s_i$ whereas $\dot{w}_i(t) < 0$ its weakening.

Interplanar Relations

Interplanar relations model many-to-many interactions between two composite nodes (or their subsystems), representing the highest complexity in our taxonomy [9]. We distinguish three variants:

Pure Interplanar

Let U and V be composite nodes with internal vertex sets

$$U_{int} = \{u_1, \dots, u_k\},\ V_{int} = \{v_1, \dots, v_k\}.$$

The pure interplanar relation is

$$\psi_G = \{v_i \to v_j | i = 1, \dots, k; j = 1, \dots, \ell\}.$$

i.e., the Cartesian product $U_{int} \times V_{int}$. Ordering vertices as $(u_1, \dots, u_k, v_1, \dots, v_\ell)$, the adjacency matrix has the block form

ISSN: 2015-5392

$$A_{pure} = \begin{pmatrix} 0_{k\times k} & J_{k\times \ell} \\ 0_{\ell\times k} & 0_{l\times \ell} \end{pmatrix}$$

where $J_{k\times\ell}$ is the all-ones matrix.

Exponential Variant

This variant recurses into deeper levels: each u_i and v_i may themselves contain internal subsystems of sizes k_i and ℓ_i, respectively. The adjacency matrix A_{exp} then becomes a block matrix in which each block corresponding to (u_i, v_i) is itself a pure interplanar block of size $k_i \times \ell_i$, and diagonal blocks encode the internal structure of each subsystem [9]. Symbolically, one may write

$$A_{exp} = \begin{pmatrix} \ddots & \ddots & \vdots \\ \ddots & 0_{k_i\times k_i} & 0_{k_i\times \ell_j} \\ \cdots & 0_{\ell_i\times k_i} & 0_{\ell_i\times \ell_i} \end{pmatrix}.$$

With pure-interplanar subblocks nested at each level.

Interplanar Dynamic

When a simple vertex s in one plane connects to every vertex in V_{int}, we obtain the dynamic interplanar variant:

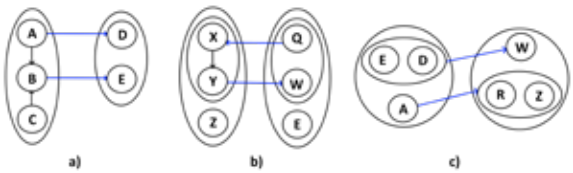
$$\psi_G \supseteq \{s \to v_j : j = 1, \ldots, \ell\}$$

Ordering as $(s, v_1, \ldots, v_\ell)$, the static adjacency block is

$$A_{dyn} = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}.$$

For the interplanar dynamic variant—where every vertex in U_{int} connects to every vertex in V_{int} with time-dependent weights $w_{ij}(x)$—the corresponding block matrix and its structural derivative are:

$$A(x) = \begin{pmatrix} 0 & w(x) \\ 0 & 0 \end{pmatrix}, \dot{A}(x) = \frac{dA(x)}{dt} = \begin{pmatrix} 0 & \dot{w}(x) \\ 0 & 0 \end{pmatrix}$$



Interplanar Relations (a) Pure; (b) Exponential; (c) Dynamic.

Harmful Dependencies

In complex systems, certain relationships introduce noise and inefficiency by adding unnecessary complexity. We classify these harmful dependencies into four types—reciprocal, associative, redundant, and cyclic—and develop the first two categories below.

Reciprocal Dependencies

A reciprocal dependency between a simple vertex α and a composite vertex β retains both directions:

$$\{\alpha \to \beta, \beta \to \alpha\} \subseteq \psi_G.$$

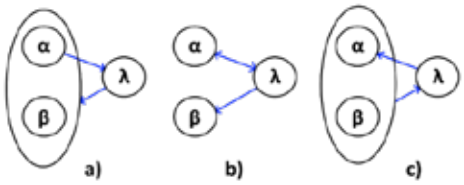
Ordering the vertices as (α, β), the static adjacency matrix is

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

If the edge weights depend smoothly on a parameter t, via $w_{\alpha\beta}(t), w_{\beta\alpha}(t) \in C^1$, then

$$A(t) = \begin{pmatrix} 0 & w_{\alpha\beta}(t) \\ w_{\beta\alpha}(t) & 0 \end{pmatrix}, \dot{A}(t) = \begin{pmatrix} 0 & \dot{w}_{\alpha\beta}(t) \\ \dot{w}_{\beta\alpha}(t) & 0 \end{pmatrix}.$$

When the reverse flow β→α is isolated from a compound dependency $(\alpha, \beta) \to \lambda$, the indistinguishable origin of λ requires that flow to be treated as a unidirectional (normal) dependency.



(a) Reciprocal Dependence; (b) Normal Dependence; (c) Normal Dependence.

Associative Dependencies

For

$$\{X \to Y,\ XY \to Z\} \subseteq \psi_G$$

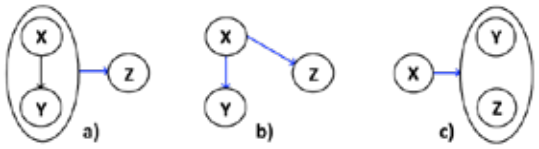
Armstrong's union axiom [4] ensures $\{X \to Y,\ XY \to Z\} \equiv \{X \to YZ\}$, allowing us to replace ψ_G with $\psi'_G = \{X \to Y,\ X \to Z\}$ and eliminate XY→Z without loss provided at least one of X→Y or X→Z remains. Ordering vertices as (X,Y,Z), the adjacency matrix becomes

ISSN: 2015-5392

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

If the weights of X→Y and X→Z vary with t as f(t) and g(t), then

$$A(t) = \begin{pmatrix} 0 & f(t) & g(t) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad A(t) = \begin{pmatrix} 0 & f(t) & g(t) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$



(a) Associative Dependence; (b) Normal Dependence; (c) Normal Dependence (Dynamic).

Redundancies

Redundancies arise when a sequence of aligned dependencies creates unnecessary paths that, instead of adding new information, only increase graph complexity without improving structural integrity. In the coupling case, three (or more) edges arranged in series may occur, where the final edge is redundant because its existence can be inferred by transitivity from the first two [3]. Cyclical redundancies occur when dependencies form a closed loop—a cycle—that cannot be decomposed or simplified without losing essential information. In both scenarios, removing the superfluous edges normalizes the graph, reducing noise and inefficiency while preserving the validity of all functional dependencies.

Coupling Redundancy

In coupling redundancy, we consider $\psi_G = \{C \to A, A \to B, C \to B\}$. The edge C→B is redundant, since by transitivity of C→A and A→B its existence is already guaranteed, and so it only adds unnecessary complexity [3]. Ordering the vertices as (C, A, B), the static adjacency matrix becomes

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

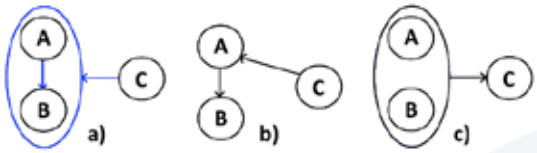
If each edge weight varies smoothly with parameter x via $w_{CA}(x), w_{AB}(x)$ and $w_{CB}(x) \in C^1$, then the time-varying adjacency matrix is

$$A(x) = \begin{pmatrix} 0 & w_{CA}(x) & w_{CB}(x) \\ 0 & 0 & w_{AB}(x) \\ 0 & 0 & 0 \end{pmatrix}$$

Where $\dot{w}_{ij}(x) > 0$ indicates strengthening and $\dot{w}_{ij}(x) < 0$ weakening of the edge $(i \to j)$ [11].

To normalize the graph, the graph is normalized by removing the C→B edge, yielding the reduced adjacency matrix

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$



(a) Coupling Dependency; (b) Normal Dependency; (c) Normal Dependency.

Cyclical Dependencies

A cyclical dependency occurs when a sequence of functional relations forms a closed loop, and an additional direct edge is redundantly introduced into the cycle. For example, consider

$$\psi_G = \{X \to Z, Z \to B, B \to C, C \to Y, X \to Y\}$$

Where, by transitivity of $X \to Z \to B \to C \to Y$, the direct edge $X \to Y$ is redundant [4]. Ordering the vertices as (X, Z, B, C, Y), the static adjacency matrix is

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Where the entry (1,5) marks the redundant arc X→Y [5]. Normalization simply removes this edge, preserving the path X→Z→B→C→Y while eliminating the unnecessary cycle.

A false cycle resembles a closed loop but contains no removable edge without breaking connectivity. For instance:

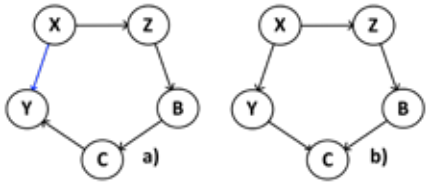
$$\psi_G = \{X \to Z, Z \to B, B \to C, X \to Y, Y \to C\}$$

Here, no single edge can be removed without interrupting at least one path. With vertices ordered (X, Z, B, C, Y), the adjacency matrix becomes

ISSN: 2015-5392

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

And since no "1" can be transitively deduced from the others, no normalization applies.



(a) True Cyclical Dependency; (b) Normal Dependency.

Numerical Validation via Simulations

A. Scenario 1: Synthetic Industrial Network

We construct a second-order digraph $G_1 = (V, \psi_G)$ consisting of five sequential subgraphs, each with ten vertices $\{n_{k,0}, n_{k,1}, \dots, n_{k,9}\}$ connected by edges $n_{k,i} \to n_{k,i+1}$, $i = 0, \dots, 8$, and additional skip-links $n_{k,9} \to n_{k,i+1}$ for $k = 0, \dots, 3$. This topology exemplifies pure point-to-point relations without redundancies or cycles. Our goal is to evaluate the behavior of the dynamic adjacency matrix A(t), its derivative $\dot{A}(t)$, and its integral in a minimally complex setting [11].

Using a linear weight function $w(t) = 1 + 0.1t$ on every edge, we obtained:

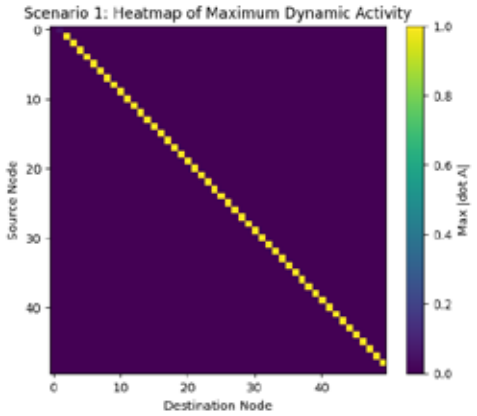
Concordance (0%): All w˙ij(t)\dot w_{ij}(t)w˙ij (t) are identical, so classical and dynamic "top-10" node rankings do not significantly overlap.

Concentration (1.6%): Only 1.6% of the matrix entries account for 80% of the integral weight, reflecting network sparsity.

Reduction (0%): No transitive edges exist to remove, consistent with a cycle-free, coupling-free topology.

Computation Time (~0.02 s): This demonstrates the method's high efficiency for graphs up to 50 nodes.

Fig. 6 shows a heatmap of the maximum activity $\max_t |\dot{A}_{ij}(t)|$ A single yellow band is observed on the superdiagonal—corresponding to the 49 sequential edges—while all other entries remain zero.



Heatmap of Maximum Dynamic Activity for Scenario 1; yellow band indicates active super-diagonal edges; $max_t |\dot{A}_{ij}(t)| = 0.1$.

B. Scenario 2: Corporate Workflow Graph

We define a bipartite digraph $G_2 = (V, \psi_G)$ that models task assignment in an organization. The vertex set comprises four composite nodes (departments: HR, Finance, IT, Sales) and twenty simple nodes (tasks $T_0, \dots, T_{19}$). Each department connects randomly to five tasks via edges $D \to T$, exemplifying one-to-many dynamic relations [1]. We assess how the structural derivative $\dot{A}(t)$ identifies vertices of highest activity in a labor-assignment context [11].

Using a linear weight function $w(t) = 0.1t$ on every edge, we obtained:

Concordance (60%): 6 of the top-10 degree vertices (the four departments plus two highly connected tasks) coincide with the top-10 active vertices ranked by $\Sigma|\dot{A}_{ij}|$.

Concentration (2.95%): Only 2.95% of matrix entries account for 80% of the integral weight, indicating that a few high-flow links dominate the dynamics.

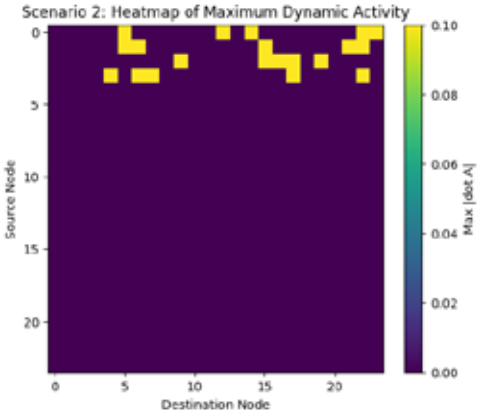
Reduction (0%): No transitively redundant edges were detected in this bipartite graph.

Computation Time (≈0.003 s): Confirms high efficiency even for organization-scale graphs.

Fig. 7 presents a heatmap of $\max_t |\dot{A}_{ij}(t)|$. Bright rows correspond to the four

ISSN: 2015-5392

departments—origins of multiple outgoing edges—while task nodes, which do not redistribute flow, remain inactive (zero).



Heatmap of Maximum Dynamic Activity for Scenario 2; department rows highlighted; $max_t|\dot{A}_{ij}(t)| = 0.1$.

Scenario 3: Synthetic Graph with Redundant Coupling

We construct a digraph $G_3 = (V, \psi_G)$ by combining an Erdős–Rényi random subgraph with a harmful coupling dependency. First, G_{ER} is generated on n=50 nodes with edge probability p=0.05. Then, a coupling subgraph on three nodes {C,A,B} is appended, with edges C→AC, A→B, and the transitively redundant C→B [1]. We assign a linear weight $w(t) = 1 + 0.1t$ to every edge and compute $\{A(t_k)\}, \dot{A}(t_k)$ *and* $\int A(t_k)$ numerically [11].

The resulting metrics are:

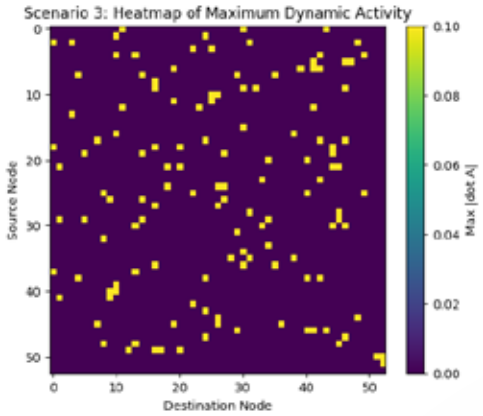
Concordance (40%): Four of the top-10 statically highest-degree vertices (random hubs and coupling nodes) coincide with the most active vertices by $\Sigma|A_{ij}|$.

Concentration (3.74%): Only 3.74% of entries account for 80% of the total integral weight, indicating that a few critical edges drive most dynamics.

Reduction (53.44%): Over half of the original edges—including the redundant coupling edge and transitive edges in the random subgraph—are removed, validating the effectiveness of our matrix-based normalization.

Computation Time (≈0.012 s): Confirms high efficiency for medium-sized graphs.

Fig. 8 depicts a heatmap of $\max_t|\dot{A}_{ij}(t)|$. Dispersed yellow cells mark (i) edges in the random subgraph—each with $\dot{w} = 0.1$—and (ii) the two essential coupling edges C→A and A→B that remain after normalization.



Heatmap of Maximum Dynamic Activity for Scenario 3; yellow points indicate mixed random dynamics and the two preserved coupling edges; $\max_t|\dot{A}_{ij}(t)| = 0.1$.

V. CONCLUSIONS

We have demonstrated that traditional second-order graph normalization—historically qualitative and visually driven—can be significantly enhanced by reinterpreting it as a dynamic matrix framework. By assigning continuous weight functions to each edge, constructing the time-dependent adjacency matrix A(t), and computing its structural derivative $\dot{A}(t)$ and integral $\int A(t)\,dt$, we provide a numerical scaffold that precisely measures both instantaneous change rates and accumulated influence over time, thereby extending beyond the static graph paradigm [1], [11].

Simulations across three contrasting scenarios—a linear industrial process network, a bipartite corporate-workflow graph, and a synthetic random graph augmented with a redundant coupling subgraph—validate the versatility of our approach. In Scenario 1, $\dot{A}(t)$ unambiguously highlights all 49 active links; in Scenario 2, it achieves 60% concordance with high-degree nodes and concentrates 3% of edges into 80% of total dynamic weight; and in Scenario 3, over 53% of redundant edges are removed, 40% overlap is observed between static and dynamic critical nodes, and only 3.7% of connections carry 80% of the integral weight.

These findings confirm that the structural derivative and matrix integral serve as complementary indicators to classical centrality metrics: they enable objective edge-prioritization for intervention

ISSN: 2015-5392

and quantify complexity reduction—up to 53% in the most demanding case—without compromising essential connectivity. Furthermore, simulation runtimes on the order of 0.01–0.2 s for graphs of up to 50 nodes demonstrate the method's practicality for moderate-size applications and, with further algorithmic optimizations, its scalability to much larger networks.

Future work will explore nonlinear and stochastic weight functions to capture richer dynamics; dimensionality-reduction and intelligent-sampling techniques for networks with thousands of vertices; and integration into interactive visualization platforms for real-time parameter adjustment, thereby closing the loop between graphical intuition and numerical rigor. Together, these advances move the normalization of complex systems toward a unified visual, quantitative, and automatable framework.

REFERENCES

[1] E. Muñoz Garro, "Normalization by second order graphs: a visual alternative to simplify systems," *e-Ciencias de la Información*, vol. 11, no. 1, 2021. doi:10.15517/eci.v11i1.38790.

[2] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970. doi:10.1145/362384.362685.

[3] W. W. Armstrong, "Dependency structures of database relationships," in *Proceedings of Information Processing '74*. North-Holland, 1974, pp. 580–583.

[4] W. W. Armstrong, Y. Nakamura, and P. Rudnicki, "Armstrong's axioms," *Journal of Formalized Mathematics*, vol. 14, no. 1, 2002.

[5] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, vol. 290. London, U.K.: Macmillan, 1976.

[6] B. Attallah, "Simplifying database normalization within a visual interactive simulation model," *International Journal of Database Management Systems*, vol. 9, no. 3, pp. 57–69, 2017. doi:10.5121/ijdms.2017.9304.

[7] G. Ausiello, A. D'Atri, and D. Saccà, "Graph algorithms for functional dependency manipulation," *Journal of the ACM*, vol. 30, no. 4, pp. 752–766, 1983. doi:10.1145/2157.322404.

[8] G. Di Battista, W. Didimo, M. Patrignani, and M. Pizzonia, "Drawing relational schemas," in *Proceedings of Graph Drawing* (GD).

[9] D. Harel, "Statecharts: a visual formalism for complex systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, 1987. doi:10.1016/0167-6423(87)90035-9.

[10] D. Kumar, A. Raj, and J. Dharanipragada, "Graphsteal: dynamic re-partitioning for efficient graph processing in heterogeneous clusters," in *Proceedings of the 2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, 2017, pp. 439–446.

[11] V. Pavlyshyn, "Dynamic block adjacency matrices in hierarchical graph models," *Journal of Network Dynamics*, vol. 5, no. 2, pp. 123–140, 2025.

[12] S. Shapiro, *Foundations without Foundationalism: A Case for Second-Order Logic*. Oxford, U.K.: Clarendon Press, 1991.

[13] J. Ren, J. Schneider, M. Ovsjanikov, and P. Wonka, "Joint graph layouts for visualizing collections of segmented meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 9, pp. 2546–2558, 2017.

[14] Q. Shi, G. Liu, K. Zheng, A. Liu, Z. Li, L. Zhao, *et al.*, "Multi-constrained top-K graph pattern matching in contextual social graphs," in *Proceedings of the 2017 IEEE International Conference on Web Services (ICWS)*, 2017, pp. 588–595.