Notes on a graph neural flow paper using Hamiltonian

Adversarial Robustness in Graph Neural Networks: A Hamiltonian Approach (K. Zhao et al., 2023, NeurIPS)

This paper aims to build a GNN that is robust to adversarial attacks by applying insights from dynamical systems/physics to the node-to-node information propagation. Intuitively, how should we enforce node message passing to follow a continuous differential equation (Graph Neural Flow) so that the GNN can perform stable inference even under attacks on graph nodes or structure? is the main research question of the paper.

- Discussion of stability from the GNN viewpoint, and inspection of stability in existing Graph Neural Flows
- Proposal of HANG (-quad), a Hamiltonian-flow-based graph neural flow that improves robustness

These notes are translated from my coursework where I had to pick a GNN-related paper and give a review. I chose this paper because I liked how it applied classic stability concepts to build more robust GNNs. As a deep learning beginner back then, I also enjoyed how it allowed me to approach the new field based on the familiar language of applied mathematics.

1. Related Work

Work 1. Neural ODE ([3]). Architectures such as RNNs, where the hidden state at a time step is expressed as a function of the previous hidden state, follow an update of the form $h_{t+1} = h_t + f(h_t, \theta_t)$. This is a discrete system where the step size between timepoints (layers) is large, and the number of timepoints is small. Neural ODEs shrink the step size to infinitesimal (conceptually) and increase the number of layers, reducing the discrete hidden-state evolution to the continuous-time ordinary differential equation $dh(t)/dt = f(h(t), t, \theta)$. This converts hidden state updates into the problem of solving differential equations, extensively studied in mathematics and physics.

Work 2. Graph Neural Flow. Researchers have started to express information propagation between adjacent nodes in a graph using Neural ODEs, interpreting a GNN as a dynamical system called graph neural flow. Some studies report that graph neural diffusion is less vulnerable to topology perturbations than conventional GNNs, but the theoretical consensus is yet to be established.

2. Stability of Graph Neural Flows

Background 1. Dynamical systems and Graph Neural Flow In this paper, the change in the system state z is defined by a function f of z and parameters θ, written as $dz/dt = f_\theta(z(t))$. From the GNN perspective, it is a continuous-time representation of how node features $z(t)$ evolve according to the graph topology (encoded in f_θ).

Background 2. Stability notions for dynamical systems and Graph Neural Flows There are multiple perspectives and definitions of stability:
Perspective 1. If a perturbation occurs at time t on node state z, does the node trajectory remain stable?

- BIBO Stability: A system is BIBO stable if for any bounded input time t and trajectory $z(t)$, there exists a constant M such that $|z(t)| < M$ for all $t \geq 0$.

- Lyapunov Stability: A system is Lyapunov stable if for every $\epsilon > 0$, there exists $\delta > 0$ such that $|z(0) - z_e| < \delta \implies |z(t) - z_e| < \epsilon$ for all $t \geq 0$. i.e. if a trajectory starts sufficiently close to an equilibrium z_e, and stays close to z_e for all future time.
- Asymptotic Stability: Lyapunov stable + $z(t)$ converges to z_e as $t \to \infty$.

Perspective 2. If the system itself is attacked/perturbed (an attack on f_θ, not on the node state) does the node trajectory remain stable?

- Structural Stability: if the graph topology (θ) is perturbed and there exists a homeomorphism that maps the perturbed trajectory $z(t)$ to the unperturbed trajectory $z(t)$, the system is structurally stable.

Perspective 3. Is there a physical quantity that remains invariant along trajectories?

- Conservative Stability: if there exists an absolute quantity that is preserved along the trajectory $z(t)$, the system is conservatively stable.

Discussion 1. Limitations of node perturbation stability Many prior works argue robustness to adversarial attacks by showing stability under node perturbations, especially Lyapunov stability. This paper argues that, from a GNN viewpoint, that alone is insufficient, through the following ODE example:

$$\dot{x}(t) = \begin{pmatrix} -1 & 0 \\ 0 & -5 \end{pmatrix} x(t)$$

Here, the map is constant over time with negative eigenvalues, meaning the trajectories converge exponentially to the equilibrium point $x(t) = 0$. Hence the system satisfies Lyapunov (and asymptotic) stability.

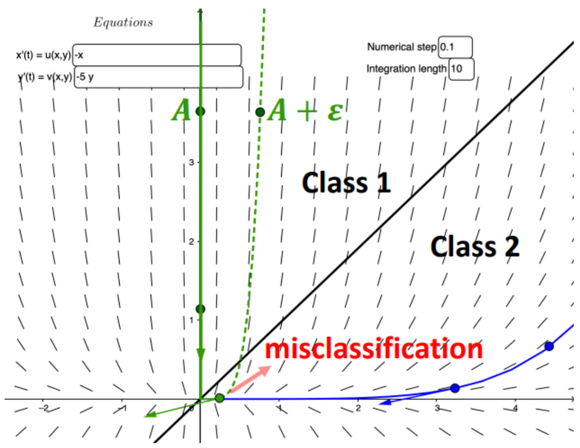
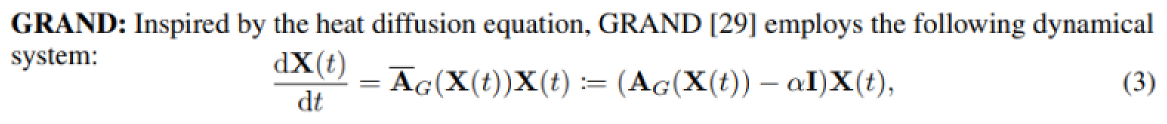

Equations
x'(t) = u(x,y) -x
y'(t) = v(x,y) -5 y
Numerical step 0.1
Integration length 10

Figure 1: Limitation of node perturbation stability

In this figure, consider a node classification task separating Class 1 and Class 2. Suppose both class trajectories share the same equilibrium at the origin and the system is Lyapunov stable. If a node A sufficiently close to the origin is perturbed to $A + \epsilon$, it may follow the green trajectory and, at some time t, be misclassified as Class 2. Even if we add structural stability, the fact that there is only one equilibrium point remains preserved, meaning the green and blue trajectories still converge to the same origin rather than distinct equilibria, and stability does not improve. Therefore, to build a more robust GNN, this paper claims that it is essential to consider whether the system has conservative stability in addition to Lyapunov (asymptotic) and structural stability.

Discussion 2. Stability of existing Graph Neural Flows The paper introduces four existing graph neural flows based on their ODE forms used for message passing, and checks whether they provide robust stability under the definitions above.

a. GRAND (using heat diffusion)


GRAND: Inspired by the heat diffusion equation, GRAND [29] employs the following dynamical system:

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = \overline{\mathbf{A}}_G(\mathbf{X}(t))\mathbf{X}(t) := (\mathbf{A}_G(\mathbf{X}(t)) - \alpha\mathbf{I})\mathbf{X}(t), \tag{3}$$

Depending on whether $A_G(X(t))$ is time-varying, GRAND is split into GRAND-nl and GRAND-1. In particular, in GRAND-nl, $A_G = a_G(x_i(t), x_j(t))$ can be interpreted as an attention matrix that computes node similarity over time.

Theorem 1. Under certain conditions, GRAND satisfies BIBO, Lyapunov, asymptotic, and conservative stability:

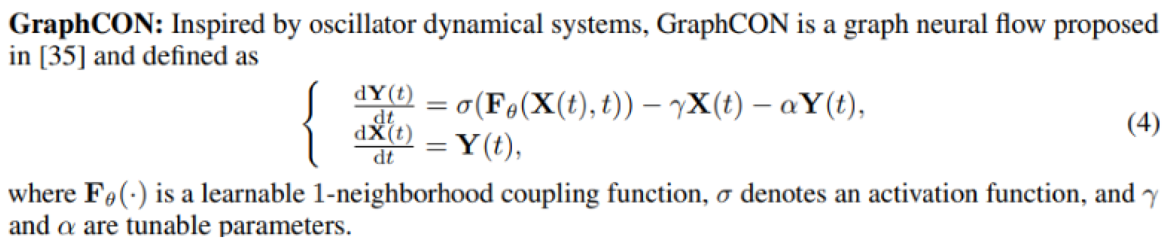
- (GRAND-nl) If $A_G(X(t))$ is doubly stochastic:
  - If $\alpha \geq 1$: BIBO, Lyapunov
  - If $\alpha > 1$: global asymptotic
- (GRAND-1) If A_G is a constant column- or row-stochastic matrix:
  - If $\alpha > 1$: global asymptotic
  - If $\alpha = 1$ and strongly connected: BIBO, Lyapunov
- (GRAND-1) If A_G is constant column-stochastic and $\alpha = 1$:
  - Conservative by default
  - If the graph is aperiodic and strongly connected, and perturbations on $X(0)$ preserve column sums: asymptotic

Intuitively, the stochasticity of A_G and the magnitude condition on α constrain the matrix driving trajectory updates so that repeated multiplicative effects do not destabilize $X(t)$.

b. BLEND: GRAND + positional encoding

If positional encoding has no effect (e.g., set to a constant), BLEND has the same structure as GRAND and inherits BIBO and Lyapunov stability under the same conditions.

c. GraphCON: oscillatory dynamics

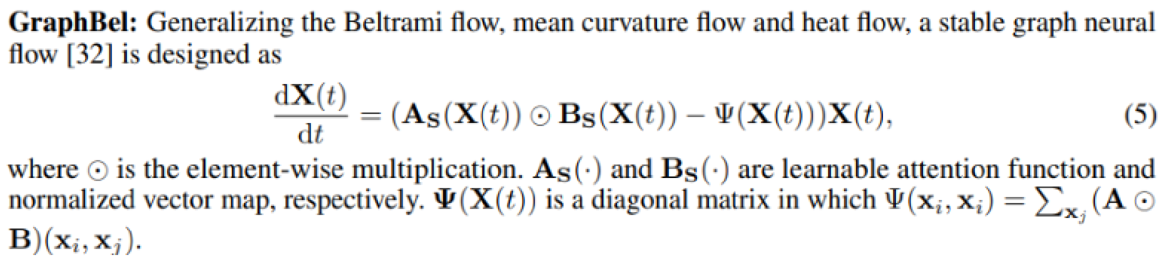

GraphCON: Inspired by oscillator dynamical systems, GraphCON is a graph neural flow proposed in [35] and defined as

$$\begin{cases} \frac{\mathrm{d}\mathbf{Y}(t)}{\mathrm{d}t} = \sigma(\mathbf{F}_\theta(\mathbf{X}(t), t)) - \gamma\mathbf{X}(t) - \alpha\mathbf{Y}(t), \\ \frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = \mathbf{Y}(t), \end{cases} \tag{4}$$

where $\mathbf{F}_\theta(\cdot)$ is a learnable 1-neighborhood coupling function, σ denotes an activation function, and γ and α are tunable parameters.

If the activation σ is identity and the coupling function $F_\theta(X(t), t)$ can be expressed as a product of a constant matrix A and $X(t)$, then the Dirichlet energy is preserved along $z(t)$, resulting in conservative stability.

d. GraphBel: Beltrami flow


GraphBel: Generalizing the Beltrami flow, mean curvature flow and heat flow, a stable graph neural flow [32] is designed as

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = (\mathbf{A}_\mathbf{S}(\mathbf{X}(t)) \odot \mathbf{B}_\mathbf{S}(\mathbf{X}(t)) - \Psi(\mathbf{X}(t)))\mathbf{X}(t), \tag{5}$$

where $\odot$ is the element-wise multiplication. $\mathbf{A}_\mathbf{S}(\cdot)$ and $\mathbf{B}_\mathbf{S}(\cdot)$ are learnable attention function and normalized vector map, respectively. $\mathbf{\Psi}(\mathbf{X}(t))$ is a diagonal matrix in which $\Psi(\mathbf{x}_i, \mathbf{x}_i) = \sum_{\mathbf{x}_j}(\mathbf{A} \odot \mathbf{B})(\mathbf{x}_i, \mathbf{x}_j)$.

If $\Psi(X(t)) = B_S(X(t)) = I$, GraphBel reduces to GRAND, and likewise has BIBO and Lyapunov stability under certain conditions.
The paper claims that most existing Graph Neural Flows satisfy BIBO/Lyapunov/conservative stability only under restrictive conditions, inspiring a design of flows that satisfy conservative stability without extra conditions.

2. Hamiltonian Flow → HANG (-quad)

Method 1. Applying Hamiltonian Flows to GNN The paper proposes to use Hamiltonian flow as a Graph Neural Flow so that the Hamiltonian value inside the GNN remains constant, improving conservative stability and robustness. In short, define per-node feature and momentum vectors, evolve them with an ODE solver following Hamilton's equations, and use the resulting node features for downstream tasks.

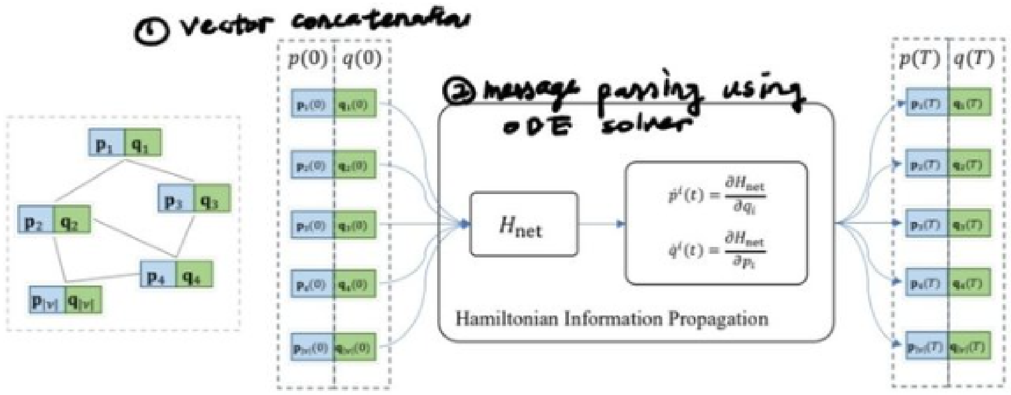

① vector concatenation
$p(0)$ $q(0)$
② message passing using ODE solver
$p(T)$ $q(T)$
$\mathbf{p}_1$ $\mathbf{q}_1$
$\mathbf{p}_2$ $\mathbf{q}_2$
$\mathbf{p}_3$ $\mathbf{q}_3$
$\mathbf{p}_4$ $\mathbf{q}_4$
$\mathbf{p}_{|\mathcal{V}|}$ $\mathbf{q}_{|\mathcal{V}|}$
H_{net}
$\dot{p}^i(t) = \frac{\partial H_{\text{net}}}{\partial q_i}$
$\dot{q}^i(t) = \frac{\partial H_{\text{net}}}{\partial p_i}$
Hamiltonian Information Propagation

Figure 2: The model architecture: each node is assigned a learnable "momentum" vector at time $t = 0$ which initializes the evolution of the system together with node features. The graph features evolve on following a *learnable* law (10) derived from the H_{net}. At the time $t = T$, we use $q(T)$ as the final node feature. $H_{\text{net}}(q(t), p(t))$ is a learnable graph energy function.

- **Vector concatenation:** At time t, define node features as an r-dimensional feature vector $q(t)$ and momentum vector $p(t)$. Concatenate all nodes' $q(t)$ and $p(t)$ to represent the whole-graph state and momentum.
- **Message passing with an ODE solver:** Starting from initial condition $(q_{\text{concat}}(0), p_{\text{concat}}(0))$, propagate information via Hamiltonian flow $\dot{q} = \partial H/\partial p$, $\dot{p} = -\partial H/\partial q$. With the initial condition and governing ODE, an ODE solver yields $(q_{\text{concat}}(t), p_{\text{concat}}(t))$ at any time t. Here H is defined as a learnable function mapping a graph input to a positive real number (specified further in Method 2).
- **Projection to extract node features:** After evolving to terminal time T, apply a projection π to $(q_{\text{concat}}(T), p_{\text{concat}}(T))$ to obtain node features $q_{\text{concat}}(T)$.
- **Downstream tasks:** Decompress $q_{\text{concat}}(T)$ into per-node $q(T)$ and use them for downstream tasks.

Intuitively (Appendix D), time in the ODE solver corresponds to the discrete depth/layers of a GNN, $q(t)$ corresponds to node features, and $p(t)$ corresponds to how node features change across layers. Having an invariant Hamiltonian H means that even as depth increases, feature values and their variations are constrained to follow a regulated evolution. Thus, unless an attack affects the entire graph across all layers, changes in $q(t)$ and $p(t)$ are regulated and mitigated.

Background 1. Stability of Hamiltonian flow By definition, using Hamiltonian flow as graph neural flow guarantees conservative stability. Moreover, by the Lagrange–Dirichlet theorem, if the Hamiltonian H is structured to satisfy certain conditions, we can also obtain Lyapunov stability. Hence, choosing H properly in method 1 can increase stability.

- Theorem 2. A graph system defined by (q, p) and a Hamiltonian $H(q, p)$ satisfying Hamiltonian flow $\dot{q} = \partial H/\partial p$, $\dot{p} = -\partial H/\partial q$ has conservative stability. Additionally, if (q, p) bounded $\implies$ $H(q, p)$ bounded and $(q, p) \to \infty \implies H(q, p) \to \infty$ it has BIBO stability.

- Theorem 3 (Lagrange–Dirichlet). Let z_e be a locally quadratic equilibrium of the Hamiltonian system. If all of the following hold, then z_e is Lyapunov stable:
  - $H = T(q,p) + U(q)$
  - $T(q,p)$ is positive definite and quadratic in p
  - z_e is a strict local minimum of $U(q)$

As a special case, energy on node features $q(t)$ can be defined as Dirichlet energy $(1/|V|)\sum_i \sum_{j \in N(i)} |q_i(t) - q_j(t)|^2$ in undirected graphs (G, V, E). Note that GraphCON evolves node features in a way that preserves this energy.

Method 2, Specifying H and proposing HANG (-quad) The paper makes the learnable Hamiltonian H concrete based on Theorems 2 and 3, to make GNN not only have conservative stability, but also Lyapunov stability.

a. Vanilla HANG (Conservative + BIBO)
Define H as the l_2-norm of the output of GCN–tanh–GCN applied to the concatenated feature and momentum vectors across all nodes. While any $H(q,p)$ yields conservative stability, graph embeddings are used to increase expressivity. (Here r is the dimension of q, p, and $|V|$ is the number of nodes.)

- Hamiltonian: $H_{\text{net}} = |(g_{\text{gcn2}} \circ \tanh \circ g_{\text{gcn1}})(q_{\text{concat}}, p_{\text{concat}})|_2$
- Layers: $g_{\text{gcn1}} : \mathbb{R}^{2r \times |V|} \to \mathbb{R}^{d \times |V|}$, $g_{\text{gcn2}} : \mathbb{R}^{d \times |V|} \to \mathbb{R}^{|V|}$

With this H, the system has conservative stability and BIBO stability by Theorem 2 (since $dH/dt = 0$, H is constant, bounded/unbounded (q,p) implies bounded/unbounded H).

b. Quadratic HANG (Conservative + BIBO + Lyapunov)
Set $H = T(q,p) + U(q)$ and enforce the Lagrange–Dirichlet conditions:

- Condition 1 (quadratic positive definite in p): $T(q,p) = \sum_k p_k^\top (A_G(q_k, q_k) A_G^\top(q_k, q_k) + \sigma I) p_k$, with $\sigma > 0$. Here A_G is a learnable adjacency/attention matrix.
- Condition 2 (strict local minima for U): $U(q) = |(\sin \circ g_{\text{gat}})(q_{\text{concat}})|_2$. Including a GAT layer increases expressivity, and sin encourages many local minima (potential Lyapunov-stable equilibria). If instead $U(q) = |\sin(q_{\text{concat}})|_2$, expressivity is too limited, elif $U(q) = |q|_2$, there are no local minima, so Lyapunov stability cannot be obtained.

Experiments

Graph Injection Attack (GIA)

- Attack algorithms: PCD-GIA, TDGIA, MetaGIA
- Datasets: citation networks (Cora, Citeseer, Pubmed), Coauthor academic network, Amazon co-purchase network, Ogbn-Arxiv
- Training/evaluation: compare node classification accuracy with/without attacks for HANG vs baseline GNNs
- Results: On citation and Coauthor academic datasets under injection/evasion/non-targeted attacks in inductive learning, HANG and HANG-quad showed higher robustness than baselines (smallest performance drop under PGD, TDGIA, MetaGIA compared to clean). Also on Amazon co-purchase and Ogbn-Arxiv under injection/evasion/targeted attacks, HANG and HANG-quad showed relatively smaller performance variation than baselines.

Graph Modification Attack

- Attack algorithm: Metaattack (+ ProGNN setting)

- Datasets: Polblogs, Pubmed
- Training/evaluation: vary attack strength (edge modification ratio from 0% to 25%) and evaluate node classification accuracy
- Results: Under modification/poisoning/non-targeted attacks in transductive learning, HANG-quad was more robust than baselines.

Additional Experiments (Appendix C)

- HANG-quad robustness was also strong under White-Box and Netattack settings
- Robustness did not vary much across different ODE solvers
- HANG/HANG-quad had higher computation time than baselines, but lower than some defense models like GCNGuard
- Combining HANG with adversarial training and GNN defense mechanisms increased robustness

Conclusion

- Stability discussion + inspection of existing Graph Neural Flows: most prior works guarantee BIBO, Lyapunov, conservative stability only under specific conditions. The paper provides an example showing that Lyapunov stability alone is insufficient to claim robustness to GNN attacks.
- HANG (-quad) via Hamiltonian flow: theoretically, HANG preserves a constant Hamiltonian $H(p, q)$ during propagation, ensuring conservative stability. HANG-quad is designed (via the Lagrange–Dirichlet theorem) to additionally obtain Lyapunov stability. Empirically, in node classification under graph injection and modification attacks, HANG and HANG-quad showed smaller performance drops than other GNNs, supporting the effect of conservative and Lyapunov stability.

References

[1] K. Zhao, Q. Kang, Y. Song, R. She, S. Wang, W. Tay, "Adversarial Robustness in Graph Neural Networks: A Hamiltonian Approach", New Orleans, USA, NeurIPS 2023

[2] I. Goodfellow, J. Shlens, C. Szegedy, "Explaining and Harnessing Adversarial Examples", Mountain View, CA, United States, ICLR 2015

[3] R. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, "Neural Ordinary Differential Equations", Montreal, Canada, NeurIPS 2018