GRAPH PROPAGATION UNDER NODE-VALUE UPDATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Propagating node values across a graph is a basic operation in graph learning. We study how to maintain the result when initial input labels change but the graph and propagation rule remain fixed. Our starting point is the Friedkin–Johnsen opinion model, whose equilibrium operator is closely related to personalized PageRank and graph neural-network propagation. We start with a variation-sensitive work bound for the residual-push framework and show that it is tight up to a constant factor on unweighted graphs. We then generalize the framework from prior work to weighted graphs through a new discounted future-work potential, and further extend it to vector-valued opinions with matrix-valued couplings. Beyond opinion dynamics, our results provide a basic model of computational rollback: when a small portion of an input is discovered to be incorrect, the output can be repaired locally instead of recomputed from scratch.

1 INTRODUCTION

Graph learning is a central topic in machine learning with applications spanning biology, network science, system optimization and trust and safety (Ju et al., 2025). In these settings, graph representations are computed by propagating neighborhood information to transform sparse, localized node signals into network-wide representations. While most dynamic graph literature assumes edge arrivals or deletions, in many real-world systems the network topology remains relatively stable while the node-level attributes and labels undergo frequent revisions. For example, in distributed or large-scale data pipelines, initial labels may be discovered to be noisy, stale, or corrupted due to sensor errors or pipeline failures, necessitating targeted point corrections without discarding downstream computation. In social and interaction networks, individual attributes, such as a user's political stance or interests, may also evolve continuously. Finally, data-governance standards and privacy mandates (e.g., the "right to be forgotten") may force systems to delete specific user-provided inputs retroactively. Because propagation operators diffuse information across multi-hop neighborhoods, even a small localized change in a node's input can create a cascading effect. Recomputing the equilibrium from scratch in modern networks is computationally expensive.

Motivated by this setting, we study the following question: what is the minimum algorithmic cost required to dynamically repair propagation or unlearn the effect of an input update? To systematically isolate this question, we study dynamic maintenance in the classical Friedkin–Johnsen (FJ) model of opinion formation (Friedkin & Johnsen, 1990). Because the FJ equilibrium operator is mathematically equivalent to Personalized PageRank and forms the backbone of decoupling-based GNNs, analyzing dynamic point updates to opinions provides a clean foundation for understanding the computational complexity of local repair and graph unlearning.

The Friedkin–Johnsen model of opinion formation (Friedkin & Johnsen, 1990) represents a social network by a graph $G = (V, E)$, whose nodes are individuals and whose edges are social connections. Each individual i has an innate opinion s_i and repeatedly revises an expressed opinion by combining this innate opinion with the expressed opinions of their neighbors. In the homogeneous model studied here, the limiting expressed-opinion vector z^* satisfies

$$z^* = \alpha s + (1 - \alpha) P z^*, \tag{1}$$

where P is the random-walk matrix of G and α measures the weight placed on innate opinions. Thus the equilibrium provides a concrete linear propagation model in which to study how a local input change affects network-wide outputs. Moreover, vector-valued extensions of Equation (1) model opinions on several interacting topics (Parsegov et al., 2016; Banihashem et al., 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Our Contribution. We study the natural dynamic version of this problem in which the graph and propagation rule are fixed, but a node's innate opinion changes—for example, because a value is refreshed or corrected.[1] Our contribution is to establish that the variation-sensitive work bound for the residual-push framework is tight and to broaden the framework's scope beyond its previous uses in PageRank and dynamic graph propagation (Andersen et al., 2006; Zhang et al., 2016; Zheng et al., 2022). After recapping the unweighted analysis in Section 3, we establish in Section 4 that its variation-sensitive work bound is tight up to a constant factor for the algorithm. We then go beyond the unweighted setting: Section 5 treats arbitrary weighted graphs using a new cost-to-go potential, and Section 6 extends this approach to vector opinions with matrix-valued couplings.

Our Techniques. To maintain an approximate equilibrium in the FJ model, we maintain an estimate–residual pair (q, r) satisfying the linear invariant $q + \alpha r = \alpha s + (1 - \alpha)Pq$, adapting the forward-push framework used in classical PageRank algorithms (Jeh & Widom, 2003; Zhang et al., 2016). We use a local push operation to propagate residuals exceeding a threshold τ, and because the equilibrium operator K is row-stochastic, ensuring $\|r\|_\infty \leq \tau$ guarantees that the maintained estimate achieves additive ℓ_∞ error at most τ. For a point update (i, Δ), the change is handled in $O(1)$ time by setting $s_i \leftarrow s_i + \Delta$ and $r_i \leftarrow r_i + \Delta$, which exactly preserves the invariant. If this injection causes $|r_i|$ to exceed τ, a cleanup phase iteratively fires local pushes, absorbing an α fraction of the residual into q and dispersing the remainder to neighboring residuals.

The key technical challenge lies in bounding the propagation work across an arbitrary sequence of updates without recomputing from scratch. In the unweighted case, the standard degree-weighted residual potential gives the variation-sensitive dynamic-forward-push bound (Zhang et al., 2016; Zheng et al., 2022); we establish that this analysis is tight for the algorithm. For weighted graphs, degree alone no longer captures the future work created as residual propagates. We therefore introduce a new discounted future-work vector $h = (I - \beta P^\top)^{-1}k$ and use the potential it induces to charge each push for its adjacency-list traversal. This potential is the key ingredient in our weighted and matrix-valued extensions.

Additional Related Work Dynamic GNNs have been extensively studied in temporal graphs, where the focus is on evolving graphs (Kumar et al., 2018; Trivedi et al., 2019; da Xu et al., 2020; Rossi et al., 2020). We refer to (Zheng et al., 2025) for a comprehensive overview. Another related line of work is the literature on dynamic *Personalized PageRank* (PPR) and linear graph propagation. Zhang et al. (2016) build on the Push framework of Andersen et al. (2006) to maintain PPR under edge updates for a randomly chosen source; in our notation, this corresponds to taking s to be a random basis vector. InstantGNN (Zheng et al., 2022) also applies dynamic push to linear GNN propagation and supports node-feature updates. Thus these works apply push-based dynamic analysis to both edge and node-feature updates. Crucially, these prior analyses assume unweighted adjacency graphs; they do not handle arbitrary edge weights, for which weighted degree governs residual flow while support degree governs computational work. We address this gap with a new cost-to-go potential, establish tightness for the unweighted push bound, and extend the analysis further to matrix-valued interactions. A separate line of work on dynamic PageRank uses random walk sampling to maintain a solution (Bahmani et al., 2010; 2012; Jayaram et al., 2024). More recently, (Jayaram et al., 2024) extend this approach to general (adversarial) updates and obtain upper and lower bounds for both additive ℓ_1 error and multiplicative error. The random-walk methods above concern updates to the graph topology, whereas our focus is on changes to the input attributes with the propagation operator held fixed.

2 PRELIMINARIES

Notation. For a vector $x \in \mathbb{R}^m$, let $\|x\|_\infty \coloneqq \max_i |x_i|$. For a matrix $M \in \mathbb{R}^{m\times n}$, let $\|M\|_\infty \coloneqq \max_i \sum_j |M_{ij}|$ denote the induced matrix infinity norm. For a square matrix M, let $\rho(M) \coloneqq \max\{|\lambda| : \lambda \text{ is an eigenvalue of } M\}$ denote its spectral radius. For a matrix M, $|M|$ denotes its entrywise absolute value. For symmetric matrices X and Y of the same dimension, $X \succeq Y$ means that $X - Y$ is positive semidefinite, while $X \succ Y$ means that $X - Y$ is positive definite.

[1] Here *dynamic* refers to changes in the input between instances and is distinct from the within-instance evolution traditionally called *opinion dynamics*.

Friedkin–Johnsen model. We first study scalar opinions on unweighted graphs, then extend the model to weighted graphs in Section 5 and to vector-valued opinions with matrix-valued couplings in Section 6. While our exposition focuses on opinion dynamics, the same equilibrium condition underlies personalized PageRank and PageRank-based graph neural networks (Jeh & Widom, 2003; Gasteiger et al., 2019; Zheng et al., 2022).

Let $G = (V, E)$ be a fixed, connected, simple, undirected graph with adjacency matrix A. For $u \in V$, let $N(u)$ denote its neighborhood and let $d_u = |N(u)|$ denote its degree. Let $D = \mathrm{diag}(d_v)$ and $P \coloneqq D^{-1}A$, so P is row-stochastic. Each node u has an innate opinion s_u and an expressed opinion $z_u^{(\ell)}$ at iteration ℓ of opinion formation. We collect these values in the vectors $s = (s_u)_{u\in V} \in \mathbb{R}^V$ and $z^{(\ell)} = (z_u^{(\ell)})_{u\in V} \in \mathbb{R}^V$. Fix a common self-confidence $\alpha \in (0, 1)$ and write $\beta \coloneqq 1 - \alpha$. Starting from an arbitrary expressed-opinion vector, the Friedkin–Johnsen process updates

$$z^{(\ell+1)} = \alpha s + \beta P z^{(\ell)}. \tag{2}$$

We note that the superscript ℓ indexes the opinion-formation process, not the dynamic input updates considered below. The process converges to the unique equilibrium z^* satisfying

$$z^* = \alpha s + \beta P z^*. \tag{3}$$

Equivalently, the equilibrium is the solution to the following system of $|V|$ linear equations:

$$z_v^* = \alpha s_v + \frac{\beta}{d_v} \sum_{u\in N(v)} z_u^*, \qquad v \in V. \tag{4}$$

Rearranging Equation (3) gives $z^* = \alpha(I - \beta P)^{-1}s$. Accordingly, define the equilibrium operator

$$K \coloneqq \alpha(I - \beta P)^{-1} = \alpha \sum_{k\geq 0} \beta^k P^k. \tag{5}$$

Thus, $z^* = Ks$. The series is well defined because $\beta < 1$ and P is row-stochastic. Moreover, every term in the series is nonnegative, and every row of K sums to $\alpha \sum_{k\geq 0} \beta^k = 1$. Hence, K is nonnegative and row-stochastic, and consequently $\|Kx\|_\infty \leq \|x\|_\infty$ for every $x \in \mathbb{R}^V$.

Dynamic problem. The graph G and parameter α remain fixed, while the innate opinion vector changes. A point update (i_t, Δ_t) replaces s_{i_t} by $s_{i_t} + \Delta_t$. After initialization and after every point update, the goal is to maintain an estimate q of the equilibrium associated with the current s such that $\|q - z^*\|_\infty \leq \tau$.

Computational model. To quantify the dependence of running time on update magnitude, we measure the graph-dependent cost by *push work*, which we define as the number of adjacency-list entries inspected or changed during residual propagation. Receiving and recording one scalar point update costs an additional $O(1)$ time per update. We account separately for initialization and for the subsequent sequence of updates. For vector-valued opinions, push work counts block operations and can understate the scalar arithmetic time because propagating residual across an edge may require a matrix–vector multiplication. We make this distinction explicit in Section 6; see Remark 6.4.

3 Warmup: push-based analysis

For completeness, we begin by recapping the push-based analysis for unweighted graphs. Local push originates in static PageRank algorithms and has subsequently been adapted to dynamic edge insertions and deletions and to node-feature updates (Andersen et al., 2006; Zhang et al., 2016; Zheng et al., 2022). This warmup introduces the invariant and potential argument that we will generalize, while Section 4 establishes that the resulting work bound is tight for the algorithm.

Algorithm and invariant. Recall from Section 2 that the equilibrium satisfies $z^* = \alpha s + \beta P z^*$ and can be written as $z^* = Ks$. Rather than recomputing the system of $|V|$ linear equations in Equation (4) after every update, the algorithm maintains an estimate q that satisfies a relaxed fixed-point equation. Specifically, it maintains a residual vector r such that

$$q_v + \alpha r_v = \alpha s_v + \frac{\beta}{d_v} \sum_{u\in N(v)} q_u, \qquad v \in V. \tag{6}$$

In matrix form, Equation (6) is $q + \alpha r = \alpha s + \beta P q$; equivalently, $z^* = q + Kr$. To see the latter equivalence, multiply $z^* - q = Kr$ by $I - \beta P$ and use $(I - \beta P)z^* = \alpha s$; the calculation is reversible. The equivalent identity $z^* = q + Kr$ provides an alternative reason to keep the residual small. In particular, because K is row-stochastic, $\|r\|_\infty \le \tau$ implies $\|q - z^*\|_\infty = \|Kr\|_\infty \le \tau$. In other words, by approximately maintaining the fixed-point equations, we can ensure that the estimate is close to the actual fixed-point solution.

This invariant can be maintained locally under a point update. If s_u changes by Δ, only the equation at u is affected, and adding Δ to r_u restores it without changing q. The residual therefore records corrections that have not yet been incorporated into the estimate and propagated through the graph. If $|r_u| > \tau$, the algorithm *pushes* u: it incorporates an α fraction of r_u into q_u, sets r_u to zero, and propagates the remaining discrepancy to the residuals of its neighbors. Only these neighbors can newly violate the threshold, so the *cleanup* procedure recursively checks them and pushes when necessary. The amount propagated to each neighbor is determined by the corresponding coordinate equation in Equation (6). If $\xi = r_u$, increasing q_u by $\alpha\xi$ and setting r_u to zero preserves the equation at u. For every $v \in N(u)$, the change to q_u increases the right-hand side of the equation for v by $\alpha\beta\xi/d_v$; increasing r_v by $\beta\xi/d_v$ therefore preserves the equation at v. The pseudocode appears in Algorithm 2 in Appendix F. It is the unweighted specialization of the weighted algorithm presented in Algorithm 1, so we omit its pseudocode here.

Theorem 3.1. *After every cleanup,*

$$\|q - z^*\|_\infty \le \tau.$$

Starting from $q = 0$, $r = s^{(0)}$, and running cleanup after initialization and after every update (i_t, Δ_t), the total push work is at most

$$\frac{1}{\alpha\tau}\left(\sum_v d_v \left|s_v^{(0)}\right| + \sum_t d_{i_t} |\Delta_t|\right).$$

The work bound is obtained by defining the degree-weighted residual potential $\Phi_{\mathrm{unw}}(r) := \frac{1}{\alpha}\sum_v d_v |r_v|$ and analyzing its change during cleanup. Intuitively, each push releases potential proportional to the degree of the pushed vertex, which is exactly its adjacency-list work. Theorem 3.1 is the special case of Theorem 5.1 in which every edge has unit weight. We therefore omit a separate proof.

4 Tightness of the dynamic push bound

We next give an unweighted instance showing that the work bound in Theorem 3.1 is tight up to a constant factor for DynamicPush.

Theorem 4.1. *For every $\alpha \in (0,1)$, threshold $\tau > 0$, and integer $d \ge 1$, set $R = \lceil 1/\alpha \rceil$ and $T = 2dR$. There is an oblivious point-update sequence of length T on $K_{d,d}$ for which* DynamicPush *performs $W = dT$ adjacency-list work, while the bound in Theorem 3.1 is at most $(9/4)W$.*

To give some intuition for Theorem 4.1, we first consider a star with one center of degree d. We repeatedly change the center's innate opinion by slightly more than τ, forcing one degree-d push after every update. Each push sends residual to the leaves, but the residual at every leaf remains below the threshold and is canceled by the next update, so no leaf is ever pushed. This example achieves work that is a factor $\Theta(\alpha)$ of the upper bound in Theorem 3.1, rather than a constant factor. Intuitively, the upper bound accounts for the residual sent to the leaves even though those leaves perform no push work.

To fix this issue, we consider a complete bipartite construction with d nodes on each side. This can be viewed as having d different star centers that share the same d leaves. After all d centers on the left have been pushed, their contributions accumulate until every vertex on the right has the residual that the left side had initially. The roles of centers and leaves are then reversed: we push the d vertices on the right, which sends the residual back to the left. Thus, after $2d$ pushes, the residuals return to their starting configuration and the process can repeat. The new construction makes the residual accumulated at the leaves trigger further pushes, so the residual charged by the upper bound corresponds to actual work. The full construction and analysis appear in Appendix A.

Algorithm 1 WEIGHTEDDYNAMICPUSH

procedure UPDATE(i, Δ)
  $s_i \leftarrow s_i + \Delta$
  $r_i \leftarrow r_i + \Delta$
  CLEANUP(i)
procedure PUSH(u)
  $\xi \leftarrow r_u$
  $q_u \leftarrow q_u + \alpha\xi$
  for all v such that $w_{vu} > 0$ **do**
    $r_v \leftarrow r_v + \beta(w_{vu}/d_v)\xi$
  $r_u \leftarrow 0$
procedure CLEANUP(u)
  if $|r_u| > \tau$ **then**
    PUSH(u)
    for all v such that $w_{vu} > 0$ **do**
      CLEANUP(v)
procedure INIT
  $q \leftarrow 0$ and $r \leftarrow s^{(0)}$
  for all $u \in V$ **do**
    CLEANUP(u)

5 WEIGHTED GRAPHS

Model and equilibrium. Let $W = (w_{uv})$ be a symmetric matrix of nonnegative edge weights, with $w_{uv} > 0$ exactly when $\{u, v\}$ is a weighted edge. Define the weighted degree and the support degree by

$$d_v \coloneqq \sum_u w_{vu}, \qquad k_v \coloneqq |\{u : w_{vu} > 0\}|,$$

and assume $d_v > 0$. Let $D = \mathrm{diag}(d_v)$ and $P = D^{-1}W$. The constant-α equilibrium remains

$$z^* = \alpha s + \beta P z^*, \qquad \text{equivalently} \qquad z_v^* = \alpha s_v + \frac{\beta}{d_v}\sum_u w_{vu} z_u^*. \tag{7}$$

The matrix P is row-stochastic, so $K = \alpha(I - \beta P)^{-1}$ is again nonnegative and row-stochastic.

Algorithm and invariant. As in the unweighted case, the algorithm maintains q and r satisfying

$$z^* = q + Kr, \qquad \text{equivalently} \qquad q + \alpha r = \alpha s + \beta P q. \tag{8}$$

The equivalence follows by the calculation after Equation (6), and the resulting local-push algorithm is the weighted analogue of the one in Section 3. The algorithm differs from the unweighted case only in the transfer coefficient and the cost of a push. Specifically, residual crosses (u, v) in proportion to w_{vu}/d_v, while scanning u costs its support degree k_u, rather than its weighted degree d_u. The full procedure is given in Algorithm 1.

Let $k = (k_v)_{v \in V}$ denote the vector of support degrees and define

$$h \coloneqq (I - \beta P^\top)^{-1} k. \tag{9}$$

Because P is stochastic, $\rho(\beta P^\top) = \beta < 1$. Moreover, the matrix $(I - \beta P^\top)^{-1} = \sum_{\ell \geq 0} \beta^\ell (P^\top)^\ell$ is coordinatewise nonnegative. Since k is also coordinatewise nonnegative, h is coordinatewise nonnegative.

The resulting guarantee for weighted graphs is as follows.

Theorem 5.1. *After every cleanup,*

$$\|q - z^*\|_\infty \leq \tau$$

Starting from $q = 0$, $r = s^{(0)}$, and running cleanup after initialization and after every update (i_t, Δ_t), the total push work is at most

$$\frac{1}{\tau}\left(\sum_v h_v \left|s_v^{(0)}\right| + \sum_t h_{i_t} |\Delta_t|\right).$$

Potential. Compared with Section 3, the weighted setting requires a new potential because the unweighted potential uses the same degree to measure both residual propagation and push cost. In a weighted graph, propagation is governed by the weighted degrees d_v and transition matrix P, whereas computational cost is governed by the support degrees k_v. Define

$$\Phi_{\mathrm{wt}}(r) \coloneqq \sum_v h_v \left| r_v \right| .$$

The coordinatewise nonnegativity of h makes Φ_{wt} a nonnegative potential, as required by the cleanup termination and telescoping arguments below.

Derivation of the potential. We choose the potential backward from the cost that it must pay. Start with unknown nonnegative weights g and the candidate

$$\Phi_g(r) \coloneqq \sum_x g_x \left| r_x \right| .$$

If a push at u processes $\xi = r_u$, then it removes $g_u \left| \xi \right|$ at u. By the triangle inequality, the residual sent to the neighbors can add at most $\beta \left| \xi \right| \sum_v (w_{vu}/d_v) g_v$ to the potential. Hence the guaranteed drop is

$$\Phi_g(r) - \Phi_g(r') \geq \left| \xi \right| \left(g_u - \beta \sum_v \frac{w_{vu}}{d_v} g_v \right) .$$

A push at u touches k_u supported edges. We therefore choose the weights so that the quantity in parentheses equals k_u, ensuring that the potential drop pays for the push. In vector form, this gives

$$g - \beta P^\top g = k.$$

Because $\beta < 1$ and P is stochastic, the inverse exists and is nonnegative, and the resulting weights are exactly $g = (I - \beta P^\top)^{-1} k = h$.

Interpretation of the potential. We start by observing that the two notions of degree play different roles. The support degree k_v is the immediate computational cost of a push at v, since the algorithm touches k_v nonzero edge entries. The weighted degree d_v, on the other hand, determines how residual flows through the weighted random walk. Thus $\frac{k_v}{d_v}$ is the local adjacency-list cost per unit of weighted residual flow.

Let $X_0, X_1, \ldots$ be the weighted random walk with transition matrix P, and put $g_v = h_v/d_v$. Symmetry of W gives $P^\top D = DP$. Multiplying both sides of the defining equation $(I - \beta P^\top)h = k$ by D^{-1} we obtain $(I - \beta P)g = D^{-1}k$. Consequently,

$$\frac{h_u}{d_u} = \mathbb{E}_u \left[\sum_{t \geq 0} \beta^t \frac{k_{X_t}}{d_{X_t}} \right] .$$

Equivalently, if L is independent and geometrically distributed with $\Pr(L = t) = \alpha \beta^t$, then

$$h_u = \frac{d_u}{\alpha} \mathbb{E}_u \left[\frac{k_{X_L}}{d_{X_L}} \right] .$$

Hence h_u is a discounted future-work weight. In an unweighted graph $k_v = d_v$ at every node, so the ratio inside the expectation is 1 and $h_u = d_u/\alpha$. Here d_u is the immediate work scale of a push at u, while $1/\alpha$ accounts for its discounted future propagation.

Proof overview. Lemma 5.2 shows that the invariant is preserved, so row-stochasticity of K gives accuracy after cleanup. Lemma 5.3 pays for one weighted push using the drop in the potential, and Lemma 5.4 sums these drops. Telescoping the potential increases caused by initialization and updates proves the work bound.

Below we state the lemmas used in the proof. Their proofs, together with the proof of Theorem 5.1, appear in Appendix B.

Lemma 5.2. *The update and push operations in Algorithm 1 preserve $z^* = q + Kr$.*

Lemma 5.3. *Suppose a weighted push at u uses the pre-push residual $\xi = r_u$. If r' is the post-push residual, then*

$$\Phi_{\mathrm{wt}}(r) - \Phi_{\mathrm{wt}}(r') \geq k_u \left|\xi\right|.$$

Lemma 5.4. *Consider one invocation of weighted cleanup, not including any preceding update. Let r^{start} be the residual upon entry to the push loop, after that update has already been applied, and let r^{end} be the residual when the loop returns. Let W_{cleanup} denote the total push work performed during this invocation. Then*

$$0 \leq W_{\mathrm{cleanup}} \leq \frac{\Phi_{\mathrm{wt}}(r^{\mathrm{start}}) - \Phi_{\mathrm{wt}}(r^{\mathrm{end}})}{\tau}.$$

Furthermore, cleanup terminates.

6 Matrix-weighted graphs with vector opinions

Model and equilibrium. Let $s_i, z_i^* \in \mathbb{R}^p$. Associate with every edge a symmetric PSD coupling matrix $W_{ij} \in \mathbb{R}^{p\times p}$ such that

$$W_{ij} = W_{ji} \succeq 0, \qquad W_{ii} = 0,$$

and define $S_i \coloneqq \sum_j W_{ij}$. Assume $S_i \succ 0$ for every node. The matrix-weighted equilibrium is

$$z_i^* = \alpha s_i + \beta S_i^{-1} \sum_j W_{ij} z_j^*. \tag{10}$$

The matrices W_{ij} need not commute or share an eigenbasis.

For the remainder of this section, stack node vectors $x_i \in \mathbb{R}^p$ as $x \in \mathbb{R}^{np}$. We use superscript parentheses only for time, so $s^{(0)}$ is the initial stacked opinion vector. We write $(x_i)_a$ for coordinate a of the block x_i. Define the matrices $\mathcal{W}, \mathcal{S}, \mathcal{T} \in \mathbb{R}^{np\times np}$ by

$$\mathcal{W} \coloneqq (W_{ij})_{i,j}, \qquad \mathcal{S} \coloneqq \mathrm{diag}(S_1, \ldots, S_n), \qquad \mathcal{T} \coloneqq \beta \mathcal{S}^{-1}\mathcal{W}.$$

Thus the (v, u) block of $\mathcal{T}$ is

$$\mathcal{T}_{vu} = \beta S_v^{-1} W_{vu}. \tag{11}$$

In stacked form, the equilibrium equation is

$$z^* = \alpha s + \mathcal{T} z^*. \tag{12}$$

The following lemma ensures that Equation (12) has a unique solution.

Lemma 6.1. *The matrix $\mathcal{T}$ satisfies $\rho(\mathcal{T}) \leq \beta < 1$. Consequently, Equation* (12) *has a unique solution.*

Proof. Since $S_i = \sum_j W_{ij}$, the equality $W_{ij} = W_{ji}$ allows us to group the two endpoint contributions of every unordered edge, giving

$$x^\top \mathcal{S} x = \sum_i x_i^\top S_i x_i = \sum_{\{i,j\}} \left(x_i^\top W_{ij} x_i + x_j^\top W_{ij} x_j \right).$$

Similarly, symmetry of the graph and $W_{ij} = W_{ji}$ imply

$$x^\top \mathcal{W} x = \sum_{i,j} x_i^\top W_{ij} x_j = 2 \sum_{\{i,j\}} x_i^\top W_{ij} x_j.$$

Subtracting and adding these two identities, respectively, yields

$$x^\top (\mathcal{S} - \mathcal{W}) x = \sum_{\{i,j\}} (x_i - x_j)^\top W_{ij} (x_i - x_j) \geq 0,$$
$$x^\top (\mathcal{S} + \mathcal{W}) x = \sum_{\{i,j\}} (x_i + x_j)^\top W_{ij} (x_i + x_j) \geq 0.$$

Hence

$$-\mathcal{S} \preceq \mathcal{W} \preceq \mathcal{S}.$$

Congruence by $\mathcal{S}^{-1/2}$ shows that every eigenvalue of $\mathcal{S}^{-1/2}\mathcal{W}\mathcal{S}^{-1/2}$ lies in $[-1, 1]$. Moreover,

$$\mathcal{S}^{-1}\mathcal{W} = \mathcal{S}^{-1/2}\left(\mathcal{S}^{-1/2}\mathcal{W}\mathcal{S}^{-1/2}\right)\mathcal{S}^{1/2},$$

so the two matrices have the same eigenvalues. Multiplication by β gives $\rho(\mathcal{T}) \leq \beta < 1$. In particular, 1 is not an eigenvalue of $\mathcal{T}$, so $I-\mathcal{T}$ is invertible. Rewriting Equation (12) as $(I-\mathcal{T})z^* = \alpha s$ therefore gives the unique solution $z^* = \alpha(I-\mathcal{T})^{-1}s$. □

Algorithm and invariant. Define

$$\mathcal{K} \coloneqq \alpha(I-\mathcal{T})^{-1}. \tag{13}$$

Then $z^* = \mathcal{K}s$. As in the preceding sections, maintain the following equivalent invariants.[2]

$$z^* = q + \mathcal{K}r, \qquad \text{equivalently} \qquad q + \alpha r = \alpha s + \mathcal{T}q. \tag{14}$$

We again use the local-push approach from the preceding sections. A point update adds the vector change Δ to both s_u and r_u. If $\|r_u\|_\infty > \tau$, a push at u stores $\xi = r_u$, adds $\alpha\xi$ to q_u, sets r_u to zero, and adds $\mathcal{T}_{vu}\xi$ to each neighboring residual r_v. Thus, the scalar transfer coefficient from the weighted setting is replaced by a matrix–vector multiplication, while cleanup still recursively checks only the neighbors affected by a push. The full pseudocode appears in Algorithm 3 in Appendix F.

Let

$$k_u \coloneqq |\{v : W_{vu} \neq 0\}|$$

be the number of block edges touched by a push at u.

Potential. Define B by taking the absolute value of $\mathcal{T}$ entrywise: $(B_{vu})_{ba} = |(\mathcal{T}_{vu})_{ba}|$. This absolute-transfer matrix is needed because matrix couplings can mix coordinates and signs. Although such cancellation may reduce the actual residual, a worst-case work bound cannot rely on it: the potential argument uses the triangle inequality and must charge every absolute contribution created by a push. The matrix B therefore describes residual propagation after all cancellation is discarded. While $\rho(\mathcal{T}) < 1$ guarantees that the equilibrium is well defined, the stronger condition $\rho(B) < 1$ ensures that even this worst-case absolute propagation decays, allowing a finite nonnegative future-work vector h. Accordingly, we make the following assumption throughout the remainder of this section.

Assumption 6.2. *The matrix B satisfies $\rho(B) < 1$.*

We verify in Section D that this assumption holds automatically in the special case of diagonal couplings.

Define $\bar{k} \in \mathbb{R}^{np}$ by assigning the block-push cost k_u to every coordinate of node u:

$$\bar{k}_{u,a} \coloneqq k_u, \qquad h \coloneqq (I - B^\top)^{-1}\bar{k}. \tag{15}$$

Since $\rho(B) < 1$, the matrix $I - B^\top$ is invertible; moreover, its Neumann-series representation shows that $(I - B^\top)^{-1}$ has nonnegative entries, so h is finite and nonnegative. Componentwise, its defining equation is

$$h_{u,a} = k_u + \sum_v \sum_{b=1}^{p} (B_{vu})_{ba} h_{v,b}. \tag{16}$$

This charges the immediate push cost k_u and all future costs generated by the residual, exactly as $h = (I - \beta P^\top)^{-1}k$ does in the weighted scalar section. Define the nodewise charge and global potential by

$$\phi_u(x) \coloneqq \sum_{a=1}^{p} h_{u,a} |x_a|, \qquad \Phi_{\mathrm{mat}}(r) \coloneqq \sum_u \phi_u(r_u). \tag{17}$$

Finally, define the exact stacked ℓ_∞ error constant

$$C_\infty \coloneqq \|\mathcal{K}\|_\infty = \max_{i,b} \sum_{j,a} |(\mathcal{K}_{ij})_{ba}|. \tag{18}$$

[2] Their equivalence is verified in Section E.

Theorem 6.3. *Under Assumption 6.2, after every cleanup,*

$$\|q - z^*\|_\infty \le C_\infty \tau. \tag{19}$$

Starting from $q = 0$, $r = s^{(0)}$, and running cleanup after initialization and after every update (i_t, Δ_t), the total push work satisfies

$$W \le \frac{1}{\tau}\left(\sum_u \phi_u(s_u^{(0)}) + \sum_{t=1}^{T} \phi_{i_t}(\Delta_t)\right). \tag{20}$$

In particular, setting $\tau = \varepsilon / C_\infty$ gives $\|q - z^*\|_\infty \le \varepsilon$.

Remark 6.4 (Block work versus arithmetic time)**.** *The work W in Theorem 6.3 uses the same convention as Theorem 5.1: receiving one update and propagating across one supported edge are each counted as $O(1)$ at the block-operation level. In scalar arithmetic, receiving the vector update Δ_t and adding it to s_{i_t} and r_{i_t} costs $O(p)$. It does not require a matrix multiplication. If the transfer matrices $\mathcal{T}_{vu}$ are precomputed, a push costs $O(p)$ to update q_u and $O(p^2)$ for the dense matrix–vector multiplication $\mathcal{T}_{vu}\xi$ on each touched block edge. Because every push touches at least one block edge, W dominates the number of pushes. Thus the dynamic arithmetic time is $O\left(pT + p^2 W\right)$, apart from preprocessing and explicit output. For constant p, this agrees with the block-level $O(T + W)$ accounting and makes the comparison with the Theorem 5.1 direct.*

Proof overview. The argument parallels Theorem 5.1, with the scalar future-work potential replaced by Φ_{mat}. Lemma 6.5 shows that the invariant is preserved, so $z^* - q = \mathcal{K}r$ and the definition of C_∞ give accuracy after cleanup. Lemma 6.6 pays for one matrix push using the drop in the potential, and Lemma 6.7 sums these drops and establishes termination. Telescoping the potential increases caused by the initial residual and subsequent updates proves the work bound.

Below we state the lemmas used in the proof. Their proofs, together with the proof of Theorem 6.3, appear in Appendix C.

Lemma 6.5. *The update and push operations in Algorithm 3 preserve $z^* = q + \mathcal{K}r$.*

Lemma 6.6. *Suppose a push at u uses the pre-push residual $\xi = r_u$. If r' is the post-push residual, then*

$$\Phi_{\mathrm{mat}}(r) - \Phi_{\mathrm{mat}}(r') \ge k_u \|\xi\|_1 \ge k_u \|\xi\|_\infty .$$

Lemma 6.7. *Consider one invocation of cleanup, not including any preceding update. Let r^{start} be the residual upon entry to the push loop, after that update has already been applied, and let r^{end} be the residual when the loop returns. Let W_{cleanup} denote the total block push work performed during this invocation. Then*

$$0 \le W_{\mathrm{cleanup}} \le \frac{\Phi_{\mathrm{mat}}(r^{\mathrm{start}}) - \Phi_{\mathrm{mat}}(r^{\mathrm{end}})}{\tau}.$$

Furthermore, cleanup terminates.

7 Conclusion

We studied how to maintain linear graph propagation when node values change but the graph remains fixed. We established that the variation-sensitive work bound for unweighted dynamic forward push is tight up to a constant factor for the algorithm. We then went beyond the predominantly unweighted prior analyses by introducing a new potential that accounts for the future push work generated by residual propagation, first for weighted graphs and then for vector-valued opinions with matrix couplings. Together, these results provide a foundation for both dynamically maintaining graph-learning computations as node values evolve and locally repairing their outputs when previously used inputs are discovered to be incorrect or corrupted, without recomputing the propagated solution from scratch.

AI use statement

In this work, we used generative AI tools to assist in the writing of proofs and, in one instance, to formulate mathematical claims and refine hypotheses (Assumption 6.2). We have not used generative AI tools to help develop theoretical models or conceptual frameworks, to formulate other

mathematical claims, to provide critical ingredients for proving mathematical claims, or to design or provide feedback on research methodology. Generating synthetic data sets, designing experiments, implementing methods, translation, cleaning or reformatting data sets, qualitative and thematic data analysis, and interpreting results are not applicable to this work. Additionally, we used generative AI tools to draft parts of the paper, identify relevant literature, and edit the paper to improve readability. We have reviewed all AI-assisted work, and the authors checked all proofs. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using PageRank vectors. In *FOCS*, 2006.

Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. Fast incremental and personalized pagerank. *Proc. VLDB Endow.*, 4(3):173–184, 2010. doi: 10.14778/1929861.1929864. URL http://www.vldb.org/pvldb/vol4/p173-bahmani.pdf.

Bahman Bahmani, Ravi Kumar, Mohammad Mahdian, and Eli Upfal. Pagerank on an evolving graph. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 24–32, 2012.

Kiarash Banihashem, MohammadTaghi Hajiaghayi, Mahdi JafariRaviz, Danny Mittal, and Alipasha Montaseri. How bad is forming your own multidimensional opinion? In Itai Ashlagi and Aaron Roth (eds.), *Proceedings of the 26th ACM Conference on Economics and Computation, EC 2025, Stanford University, Stanford, CA, USA, July 7-10, 2025*, pp. 1076–1104. ACM, 2025. doi: 10.1145/3736252.3742669. URL https://doi.org/10.1145/3736252.3742669.

da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJeW1yHYwH.

Noah E. Friedkin and Eugene C. Johnsen. Social influence and opinions. *The Journal of Mathematical Sociology*, 15(3–4):193–206, 1990.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized PageRank. In *ICLR*, 2019.

Rajesh Jayaram, Jakub Lacki, Slobodan Mitrovic, Krzysztof Onak, and Piotr Sankowski. Dynamic pagerank: Algorithms and lower bounds. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson (eds.), *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, Tallinn, Estonia, July 8-12, 2024*, LIPIcs, pp. 90:1–90:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi: 10.4230/LIPICS.ICALP.2024.90. URL https://doi.org/10.4230/LIPIcs.ICALP.2024.90.

Glen Jeh and Jennifer Widom. Scaling personalized web search. In *WWW*, 2003.

Wei Ju, Siyu Yi, Yifan Wang, Zhiping Xiao, Zhengyang Mao, Hourun Li, Yiyang Gu, Yifang Qin, Nan Yin, Senzhang Wang, et al. A survey of graph neural networks in real world: Imbalance, noise, privacy and ood challenges. *IEEE transactions on pattern analysis and machine intelligence*, 2025.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Learning dynamic embeddings from temporal interactions. *arXiv preprint arXiv:1812.02289*, 2018.

Sergey E Parsegov, Anton V Proskurnikov, Roberto Tempo, and Noah E Friedkin. Novel multidimensional models of opinion dynamics in social networks. *IEEE Transactions on Automatic Control*, 62(5):2270–2285, 2016.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019.

Hongyang Zhang, Peter Lofgren, and Ashish Goel. Approximate personalized pagerank on dynamic graphs. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1315–1324, 2016.

Yanping Zheng, Hanzhi Wang, Zhewei Wei, Jiajun Liu, and Sibo Wang. Instant graph neural networks for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 2605–2615, 2022.

Yanping Zheng, Lu Yi, and Zhewei Wei. A survey of dynamic graph neural networks. *Frontiers of Computer Science*, 19(6):196323, 2025.

APPENDICES

A Proof for Section 4 13
A.1 Proof of Theorem 4.1 13
B Proofs for Section 5 15
B.1 Proof of Lemma 5.2 15
B.2 Proof of Lemma 5.3 15
B.3 Proof of Lemma 5.4 15
B.4 Proof of Theorem 5.1 15
C Proofs for Section 6 16
C.1 Proof of Lemma 6.5 16
C.2 Proof of Lemma 6.6 16
C.3 Proof of Lemma 6.7 16
C.4 Proof of Theorem 6.3 17
D Special case: diagonal couplings 17
E Equivalence of the matrix invariants 18
F Pseudocode 19
F.1 Unweighted dynamic push 19
F.2 Matrix dynamic push 19

A PROOF FOR SECTION 4

A.1 PROOF OF THEOREM 4.1

Proof. We first consider a simpler construction and explain why it does not quite prove the desired tightness. Let G be a star with center c of degree d, initialize $s = q = r = 0$, and alternately change s_c from 0 to ℓ, from ℓ back to 0, and so on, where $\ell = (1+\alpha/2)\tau$. Each update changes r_c by $+\ell$ or $-\ell$. Since $\ell > \tau$, cleanup pushes the center at cost d. That push resets r_c to zero and sends a residual of $\pm\beta\ell$ to each leaf, because the receiving leaf has degree one. The next, oppositely signed center push cancels this leaf residual, and no leaf is ever pushed because $\beta\ell = (1-\alpha)(1+\alpha/2)\tau < \tau$. Thus cleanup performs exactly one center push per update, and over T updates the actual adjacency-list work is

$$W_{\mathrm{star}} = dT.$$

This example shows that a single point update can force degree-proportional work. It does not, however, make the bound in Theorem 3.1 tight uniformly in α: every flip has magnitude $\ell = \Theta(\tau)$, so the update term in that bound is

$$U_{\tau,\mathrm{star}} = \frac{dT\ell}{\alpha\tau} = \Theta\left(\frac{dT}{\alpha}\right),$$

which can exceed the actual work by a factor $\Theta(1/\alpha)$. The construction below instead uses a complete bipartite graph. One group of d vertices plays the role of centers whose pushes accumulate residual at the other group; once that residual has accumulated, the roles reverse and the residual is pushed back. This recycling allows each degree-d push to be triggered by an update of only $\Theta(\alpha\tau)$.

By Theorem 3.1, the adjacency-list work W performed by DYNAMICPUSH satisfies

$$W \le U_\tau \coloneqq \frac{\sum_v d_v \left|s_v^{(0)}\right| + \sum_t d_{i_t} |\Delta_t|}{\alpha\tau}. \tag{21}$$

We now give an update sequence on which the actual work is within a constant factor of this upper bound.

Here is the exact adversarial sequence. It is oblivious: all updates can be fixed before the algorithm runs. Let $G = K_{d,d}$ with bipartition $X = \{x_1, \dots, x_d\}$ and $Y = \{y_1, \dots, y_d\}$, and write $\beta = 1-\alpha$. Set

$$\ell = (1 + \alpha/2)\tau.$$

Then

$$\ell > \tau, \qquad \beta\ell = (1-\alpha)(1+\alpha/2)\tau < \tau.$$

Initialize $q = 0$ and

$$r_x = s_x^{(0)} = \beta\ell \quad (x \in X), \qquad r_y = s_y^{(0)} = 0 \quad (y \in Y).$$

This initialization satisfies the system Equation (6). Moreover, $\|r\|_\infty = \beta\ell < \tau$, so initialization performs no push.

Now process $x_1, \dots, x_d$ in order. At x_j, make the point update

$$s_{x_j} \leftarrow s_{x_j} + \alpha\ell, \qquad r_{x_j} \leftarrow r_{x_j} + \alpha\ell.$$

Before this update, $r_{x_j} = \beta\ell$; afterward it equals $\ell > \tau$. Thus x_j is pushed, at cost d, and sends $\beta\ell/d$ residual to every vertex of Y. After the first k such pushes,

$$r_y = \frac{k\beta\ell}{d} \le \beta\ell < \tau \qquad (y \in Y).$$

Processed vertices of X have residual zero, and unprocessed vertices still have residual $\beta\ell < \tau$. Hence the active x_j is the only threshold violation and cleanup performs exactly one push.

After all d vertices of X have been processed, the residual is zero on X and $\beta\ell$ on Y. Processing $y_1, \dots, y_d$ by the same updates and pushes restores residual $\beta\ell$ on X and zero on Y. One complete round therefore has the residual evolution

$$(\beta\ell \text{ on } X,\ 0 \text{ on } Y) \longrightarrow (0 \text{ on } X,\ \beta\ell \text{ on } Y) \longrightarrow (\beta\ell \text{ on } X,\ 0 \text{ on } Y).$$

Thus the residual returns to its initial state, although s and q have changed. Repeating the two waves for R rounds gives

$$T = 2dR, \qquad W = dT.$$

The construction works because each wave supplies the next side with $\beta\ell$ residual, so the adversary needs to add only $\alpha\ell = \Theta(\alpha\tau)$ to force each degree-d push.

For this sequence, the two terms in the upper bound Equation (21) are

$$\sum_v d_v \left|s_v^{(0)}\right| = d^2\beta\ell, \qquad \sum_t d_{i_t} |\Delta_t| = dT\alpha\ell.$$

Consequently, Theorem 3.1 gives the concrete upper bound

$$U_\tau = \frac{d^2\beta\ell}{\alpha\tau} + \frac{dT\ell}{\tau}.$$

With $R = \lceil 1/\alpha \rceil$ and $T = 2dR$,

$$\frac{d^2\beta\ell/(\alpha\tau)}{dT\ell/\tau} = \frac{\beta}{2\alpha R} \le \frac{1}{2}.$$

Also $1 < \ell/\tau \le 3/2$. Therefore

$$W = dT \le U_\tau \le \frac{9}{4}dT = \frac{9}{4}W.$$

It follows that U_τ is within a factor $9/4$ of W, as claimed.

□

B Proofs for Section 5

B.1 Proof of Lemma 5.2

Proof. An update changes s_i and r_i by the same Δ, preserving the local invariant. For a push at u, let $\xi = r_u$ be the pre-push residual. The estimate and residual after the push are

$$q' = q + \alpha\xi e_u,$$
$$r' = r - \xi e_u + \beta\xi \sum_{v:w_{vu}>0} \frac{w_{vu}}{d_v} e_v.$$

We have

$$Pe_u = \sum_v \frac{w_{vu}}{d_v} e_v.$$

Substituting these expressions into Equation (8) shows that the invariant is preserved. □

B.2 Proof of Lemma 5.3

Proof. The definition of h gives

$$h_u - \beta \sum_v \frac{w_{vu}}{d_v} h_v = k_u.$$

The push removes $h_u |\xi|$ and increases the neighbor terms by at most $\beta |\xi| \sum_v \frac{w_{vu}}{d_v} h_v$. Subtracting the two terms, the total potential drop is at least $k_u |\xi|$. □

B.3 Proof of Lemma 5.4

Proof. If no coordinate of r^{start} violates the threshold, then $W_{\text{cleanup}} = 0$ and $r^{\text{end}} = r^{\text{start}}$, so the inequality holds with equality.

Otherwise, every pushed node has $|r_u| > \tau$. By Lemma 5.3, its push decreases the potential by more than τk_u, paying for the k_u supported edges it touches. Summing over pushes gives the bound. Each potential drop is at least $\tau k_u > 0$, and the potential is nonnegative, so cleanup terminates. □

B.4 Proof of Theorem 5.1

Proof. By Lemma 5.2, the invariant is preserved and $z^* - q = Kr$. Since K is row-stochastic, it follows that $\|z^* - q\|_\infty = \|Kr\|_\infty \le \|r\|_\infty \le \tau$, where the last inequality follows from the definition of cleanup.

It remains to account for the total push work W. The initial residual has potential

$$\Phi_{\text{wt}}(s^{(0)}) = \sum_v h_v \left| s_v^{(0)} \right|.$$

An update at i_t increases the potential by at most

$$\Phi_{\text{wt}}(r + \Delta_t e_{i_t}) - \Phi_{\text{wt}}(r) \le h_{i_t} |\Delta_t|.$$

Let $a_0 = s^{(0)}$ be the residual before the initial cleanup and b_0 the residual afterward. For $t \ge 1$, let $a_t = b_{t-1} + \Delta_t e_{i_t}$ be the residual after update t, and let b_t be the residual after its cleanup. By Lemma 5.4,

$$\begin{aligned}
\tau W &\le \sum_{t=0}^{T} \left(\Phi_{\text{wt}}(a_t) - \Phi_{\text{wt}}(b_t)\right) \\
&= \Phi_{\text{wt}}(a_0) + \sum_{t=1}^{T} \left(\Phi_{\text{wt}}(a_t) - \Phi_{\text{wt}}(b_{t-1})\right) - \Phi_{\text{wt}}(b_T) \\
&\le \sum_v h_v \left| s_v^{(0)} \right| + \sum_{t=1}^{T} h_{i_t} |\Delta_t|.
\end{aligned}$$

Dividing by τ proves the work bound. □

C Proofs for Section 6

C.1 Proof of Lemma 6.5

Proof. An update changes s_{i_t} and r_{i_t} by the same Δ_t, so it preserves the local invariant in Equation (14). During a push at u, the change in the uth block of $q + \alpha r$ is

$$\alpha\xi - \alpha\xi = 0.$$

For every $v \neq u$, its change is

$$\alpha\mathcal{T}_{vu}\xi.$$

On the right-hand side, only q_u changes, by $\alpha\xi$, so the change in the vth block of $\mathcal{T}q$ is likewise $\alpha\mathcal{T}_{vu}\xi$. Thus every block of the local invariant is preserved, and hence so is the equivalent global invariant. □

C.2 Proof of Lemma 6.6

Proof. Let y be the stacked vector with block $y_u = \xi$ and all other blocks zero. A push gives

$$r' = r - y + \mathcal{T}y.$$

For a stacked vector x, let $|x|$ denote coordinatewise absolute value; that is,

$$(|x|)_{v,a} \coloneqq |(x_v)_a| \,.$$

Inequalities between stacked vectors in this proof are understood coordinatewise. Since $B = |\mathcal{T}|$, the triangle inequality gives for every node v and coordinate a,

$$\begin{aligned} |(r'_v)_a| &= \left| (r_v - y_v)_a + \sum_{w,b} (\mathcal{T}_{vw})_{ab} (y_w)_b \right| \\ &\leq |(r_v - y_v)_a| + \sum_{w,b} |(\mathcal{T}_{vw})_{ab}| \, |(y_w)_b| \,. \end{aligned}$$

Equivalently,

$$|r'| \leq |r - y| + B\,|y| \,.$$

Moreover, $r - y$ and y have disjoint block supports, so $|r| = |r - y| + |y|$. Recalling that $\Phi_{\mathrm{mat}}(r) = h^\top |r|$, we obtain

$$\begin{aligned} \Phi_{\mathrm{mat}}(r) - \Phi_{\mathrm{mat}}(r') &\geq h^\top |y| - h^\top B\,|y| \\ &= \left(h - B^\top h\right)^\top |y| \\ &= \bar{k}^\top |y| \\ &= \sum_a k_u \, |\xi_a| \\ &= k_u \, \|\xi\|_1 \geq k_u \, \|\xi\|_\infty \,, \end{aligned}$$

where the third line uses $h - B^\top h = \bar{k}$ from Equation (15). □

C.3 Proof of Lemma 6.7

Proof. If no block of r^{start} violates the threshold, then $W_{\mathrm{cleanup}} = 0$ and $r^{\mathrm{end}} = r^{\mathrm{start}}$, so the inequality holds with equality.

Otherwise, every pushed block satisfies $\|r_u\|_\infty > \tau$. Lemma 6.6 therefore releases more than τk_u potential and pays for the k_u touched block edges. Summing proves the bound. Nonnegativity of the potential proves termination. □

C.4 Proof of Theorem 6.3

Proof. The invariant gives

$$z^* - q = \mathcal{K} r.$$

After cleanup, $\|r\|_\infty \le \tau$, so

$$\|q - z^*\|_\infty \le \|\mathcal{K}\|_\infty \|r\|_\infty \le C_\infty \tau.$$

The initial potential is

$$\Phi_{\mathrm{mat}}(s^{(0)}) = \sum_u \phi_u(s_u^{(0)}).$$

Let r and r' be the residuals immediately before and after update t. Only block i_t changes. Since $h_{i_t,a} \ge 0$, the triangle inequality gives

$$\begin{aligned}\Phi_{\mathrm{mat}}(r') - \Phi_{\mathrm{mat}}(r) &= \sum_a h_{i_t,a}\big(|(r_{i_t})_a + (\Delta_t)_a| - |(r_{i_t})_a|\big) \\ &\le \sum_a h_{i_t,a}\,|(\Delta_t)_a| = \phi_{i_t}(\Delta_t).\end{aligned}$$

Applying Lemma 6.7 to every cleanup and then using the same telescoping argument as in the proof of Theorem 5.1 proves Equation (20). □

D Special case: diagonal couplings

We use square-bracket superscripts to index coordinates. In particular, $w_{ij}^{[a]}$ denotes the coupling weight between i and j in coordinate $a \in \{1, \dots, p\}$.

We now verify Assumption 6.2 when the coupling matrices are diagonal in the given coordinates:

$$W_{ij} = \mathrm{diag}\left(w_{ij}^{[1]}, \dots, w_{ij}^{[p]}\right). \tag{22}$$

This is the limited-interaction case: residual propagates between nodes but never between opinion coordinates.

For each coordinate a, define

$$d_i^{[a]} := \sum_j w_{ij}^{[a]}, \qquad P_{ij}^{[a]} := \frac{w_{ij}^{[a]}}{d_i^{[a]}}.$$

Assume $d_i^{[a]} > 0$ for every i and a. Then each $P^{[a]}$ is row-stochastic, and

$$\mathcal{T}_{vu} = \mathrm{diag}\left(\beta P_{vu}^{[1]}, \dots, \beta P_{vu}^{[p]}\right). \tag{23}$$

All entries in Equation (23) are nonnegative, so $B = |\mathcal{T}| = \mathcal{T}$. Moreover, each coordinate subspace is invariant under B, and its restriction to coordinate a is $\beta P^{[a]}$. Therefore

$$\rho(B) = \max_a \rho\left(\beta P^{[a]}\right) = \beta < 1. \tag{24}$$

Thus diagonal couplings satisfy Assumption 6.2 automatically.

The future-work weights from Equation (15) simplify to

$$h_{\cdot,a} = \left(I - \beta (P^{[a]})^\top\right)^{-1} k. \tag{25}$$

Hence Theorem 6.3 uses the nodewise update charge

$$\phi_u(x) = \sum_{a=1}^{p} h_{u,a}\,|x_a|. \tag{26}$$

Each $h_{\cdot,a}$ has the future-work form from Section 5, using the weights of coordinate a but the block support degree k. The equilibrium operator is nonnegative and row-stochastic coordinatewise, so $C_\infty = 1$. Consequently, the theorem gives $\|q - z^*\|_\infty \le \tau$ and

$$W \le \frac{1}{\tau}\left(\sum_u \phi_u(s_u^{(0)}) + \sum_{t=1}^{T} \phi_{i_t}(\Delta_t)\right). \tag{27}$$

Diagonal structure also reduces the arithmetic cost. Multiplying $\mathcal{T}_{vu}$ by a residual vector costs $O(p)$ rather than $O(p^2)$. If P_{push} pushes occur, the dynamic arithmetic time is therefore

$$O\left(p\left(T + P_{\mathrm{push}} + W\right)\right),$$

apart from preprocessing and explicit output.

For comparison, apply Theorem 5.1 separately to each coordinate. For coordinate a, define its support degree and future-work vector by

$$k_u^{[a]} := \left|\{v : w_{vu}^{[a]} > 0\}\right|, \qquad \widetilde{h}^{[a]} := \left(I - \beta(P^{[a]})^\top\right)^{-1} k^{[a]}.$$

If $W^{[a]}$ denotes the scalar edge work performed for coordinate a, Theorem 5.1 gives

$$W^{[a]} \le \frac{1}{\tau}\left(\sum_u \widetilde{h}_u^{[a]} \left|(s_u^{(0)})_a\right| + \sum_{t=1}^{T} \widetilde{h}_{i_t}^{[a]} \left|(\Delta_t)_a\right|\right). \tag{28}$$

Summing Equation (28) over the coordinates therefore yields

$$\sum_{a=1}^{p} W^{[a]} \le \frac{1}{\tau}\left(\sum_u \sum_{a=1}^{p} \widetilde{h}_u^{[a]} \left|(s_u^{(0)})_a\right| + \sum_{t=1}^{T} \sum_{a=1}^{p} \widetilde{h}_{i_t}^{[a]} \left|(\Delta_t)_a\right|\right). \tag{29}$$

When every coordinate has the same edge support, $k^{[a]} = k$ and $\widetilde{h}_u^{[a]} = h_{u,a}$. Hence the right-hand side of Equation (29) is exactly the right-hand side of the matrix bound in Equation (27). The two bounds nevertheless control different units of work. The left-hand side of Equation (29) counts individual scalar edge-coordinate operations, whereas W in Equation (27) counts an entire block-edge operation as one unit. A diagonal block-edge operation performs p scalar coordinate operations, so converting the matrix bound to the scalar-operation model introduces an additional factor of p. The coordinatewise bound avoids this factor and is therefore sharper in that model.

E EQUIVALENCE OF THE MATRIX INVARIANTS

The equilibrium equation Equation (12) gives

$$(I - \mathcal{T})z^* = \alpha s.$$

If the local invariant in Equation (14) holds, then

$$(I - \mathcal{T})q + \alpha r = \alpha s = (I - \mathcal{T})z^*.$$

Therefore

$$z^* - q = \alpha(I - \mathcal{T})^{-1} r = \mathcal{K} r,$$

which is the global invariant. Reversing these equalities proves the converse.

F PSEUDOCODE

F.1 UNWEIGHTED DYNAMIC PUSH

Algorithm 2 DYNAMICPUSH

procedure UPDATE(i, Δ)
  $s_i \leftarrow s_i + \Delta$
  $r_i \leftarrow r_i + \Delta$
  CLEANUP(i)
procedure PUSH(u)
  $\xi \leftarrow r_u$
  $q_u \leftarrow q_u + \alpha\xi$
  for all $v \in N(u)$ **do**
    $r_v \leftarrow r_v + \beta\xi/d_v$
  $r_u \leftarrow 0$
procedure CLEANUP(u)
  if $|r_u| > \tau$ **then**
    PUSH(u)
    for all $v \in N(u)$ **do**
      CLEANUP(v)
procedure INIT
  $q \leftarrow 0$ and $r \leftarrow s^{(0)}$
  for all $u \in V$ **do**
    CLEANUP(u)

F.2 MATRIX DYNAMIC PUSH

Algorithm 3 MATRIXDYNAMICPUSH

procedure UPDATE(i, Δ)
  $s_i \leftarrow s_i + \Delta$
  $r_i \leftarrow r_i + \Delta$
  CLEANUP(i)
procedure PUSH(u)
  $\xi \leftarrow r_u$
  $q_u \leftarrow q_u + \alpha\xi$
  for all v such that $W_{vu} \neq 0$ **do**
    $r_v \leftarrow r_v + \mathcal{T}_{vu}\xi$
  $r_u \leftarrow 0$
procedure CLEANUP(u)
  if $\|r_u\|_\infty > \tau$ **then**
    PUSH(u)
    for all v such that $W_{vu} \neq 0$ **do**
      CLEANUP(v)
procedure INIT
  $q \leftarrow 0$ and $r \leftarrow s^{(0)}$
  for all $u \in V$ **do**
    CLEANUP(u)