ARTICLE

A Streaming Graph Embedding Method with Temporal Regularization and Cluster-Wide Consistency Enforcement

Antonio Serrano,[1] Rogelio Manansala,[2] and Eduardo Dominguez[3]

[1]San Lorenzo Polytechnic University, Department of Computer Science and Engineering, Governor Santos Avenue, San Pedro City 4023, Philippines

[2]Western Archipelago Institute of Technology, Department of Computer Science and Engineering, Mabini Extension Road, Calapan City 5200, Philippines

[3]Nueva Aurora State College, Department of Computer Science and Engineering, Lapu-Lapu Coastal Highway, Roxas City 5800, Philippines

Abstract

Graphs that evolve through edge arrivals, deletions, and node churn are increasingly common in transactional, communication, and interaction logs, yet many embedding pipelines still assume that the full graph is available and stationary. Streaming settings introduce additional constraints: updates must be processed with limited memory, the representation should remain temporally stable to preserve downstream semantics, and local updates should not cause inconsistent drift across groups of nodes that behave as coherent communities. This paper develops a streaming graph embedding method that couples temporal regularization with an explicit cluster-wide consistency mechanism. The method maintains node embeddings under a continuously arriving stream of edges by optimizing a reconstruction objective augmented with a temporal smoothness term that penalizes abrupt representation changes across successive time windows. To mitigate representation fragmentation, a cluster-level latent structure is maintained online; embeddings are encouraged to remain consistent with evolving cluster centroids, and centroid dynamics are themselves regularized to avoid instability. The resulting objective admits efficient incremental optimization using stochastic updates and lightweight cluster maintenance, including principled handling of node arrivals and disappearances. The approach is described in a unified formulation that supports both directed and undirected graphs, accommodates negative sampling for scalability, and provides explicit controls for forgetting and adaptation. Analytical discussion focuses on stability, sensitivity to nonstationarity, and computational complexity under realistic streaming constraints, alongside a detailed treatment of practical design choices.

1. Introduction

Learning low-dimensional representations of graph vertices is a common strategy for enabling prediction, retrieval, and anomaly detection on relational data [1]. In many operational environments, however, graphs are not static artifacts but streams generated by user actions, sensor events, financial transfers, or system telemetry. The embedding problem then becomes intrinsically online: at time t the learner observes a partial and possibly transient view of the graph, processes new events, and must update representations before the next batch arrives. A purely batch approach that re-embeds the entire graph at each time step can be computationally prohibitive and may produce embeddings that rotate or drift in ways that are not aligned with semantic change [2]. In addition, even when a streaming optimizer is used, node embeddings can evolve incoherently, with nodes from the same functional group separating due to the stochasticity of incremental updates. Such group fragmentation can degrade tasks like community-aware recommendation, cluster-conditioned forecasting, and entity resolution, where relative geometry within and across groups is important.

Two desiderata are therefore emphasized [3]. The first is temporal stability: embeddings should not change more than necessary between consecutive time windows, yet must still be flexible enough to adapt when the underlying graph structure changes. This is not merely an aesthetic property; temporal stability can reduce the need for frequent retraining of downstream models and can simplify interpretation. The second is cluster-wide consistency: if there exists a latent grouping structure that persists over time, then representation updates should preserve that structure in a controlled way. In streaming graphs, local edge updates can disproportionately affect particular nodes, creating representation drift that is inconsistent with the behavior of the group [4]. A group-level mechanism can act as a stabilizer that shares statistical strength and discourages spurious divergence.

This paper proposes a streaming graph embedding method with temporal regularization and cluster-wide consistency enforcement. The method maintains node embeddings that are updated from an edge stream using a reconstruction objective with negative sampling and a time-decayed temporal penalty [5]. In parallel, it maintains a set of evolving cluster centroids and soft assignments that are updated online. The embeddings are regularized toward cluster centroids, and centroids are regularized across time, thereby enforcing consistency at two levels: node-to-previous-self and node-to-cluster. The combination is intended to address both within-node stability and within-cluster coherence without requiring expensive global re-optimization [6].

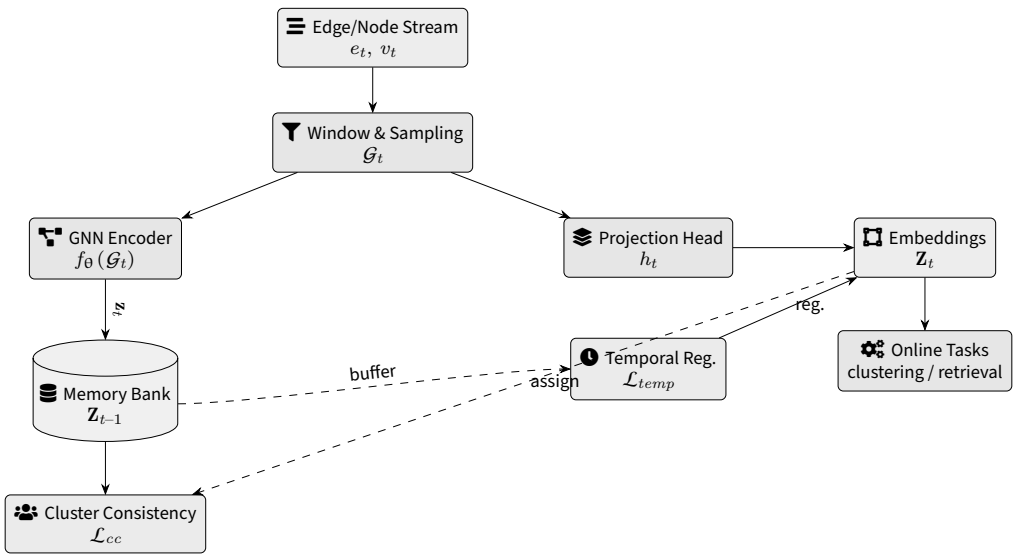


Figure 1. System-level flow for streaming graph embedding. Each time step builds a windowed graph $\mathcal{G}_t$, encodes it to produce embeddings $\mathbf{Z}_t$, and applies temporal regularization using stored history $\mathbf{Z}_{t-1}$ while enforcing cluster-wide consistency through a shared objective.

The core contribution is not a particular reconstruction loss but a modular formulation that can be instantiated with common link-based objectives, together with an optimization strategy that remains feasible under streaming constraints. The formulation makes explicit the tension among adaptability, stability, and consistency, offering interpretable hyperparameters that govern forgetting rates, cluster inertia, and regularization strength. The subsequent sections formalize the streaming setting and notation, define the objective, derive update rules and implementation details, analyze stability and complexity, and discuss empirical evalua-

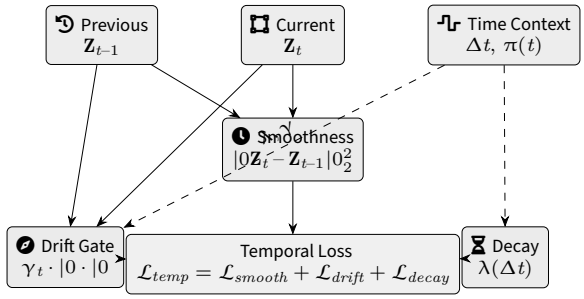


Figure 2. Temporal regularization for streaming embeddings. A smoothness term couples adjacent snapshots, a drift gate modulates penalties under concept shift, and a time-aware decay scales constraints as the interval Δt grows.

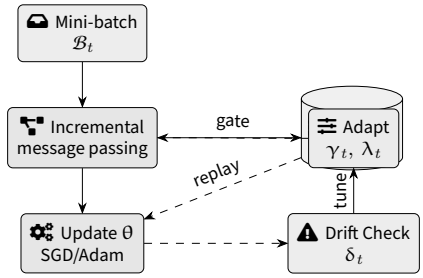


Figure 3. Online training loop. Each stream batch triggers incremental propagation and parameter updates while a memory bank enables replay; a drift score δ_t adjusts regularization gates to balance stability and adaptation.

tion considerations and expected behaviors under different types of graph nonstationarity.

2. Problem Setting and Notation

Consider a dynamic graph observed through an event stream [7]. At time t a learner observes a set of interactions that can be represented as a multiset of directed or undirected edges. To accommodate both event-level and windowed processing, let $\mathcal{B}_t$ denote the batch of events in a time window indexed by t, and let $G_t = (V_t, E_t)$ denote the aggregated graph state after processing $\mathcal{B}_t$, where V_t is the active node set and E_t is the multiset or set of edges up to time t subject to a retention policy. The retention policy may be cumulative, sliding-window, or exponential decay. The method described here is compatible with these variants by incorporating a time-decay mechanism into the objective [8].

Let d be the embedding dimension. Each node $i \in V_t$ is represented by an embedding vector $x_{i,t} \in \mathbb{R}^d$. Collect embeddings into a matrix $X_t \in \mathbb{R}^{|V_t| \times d}$. For directed graphs, it is often useful to use two embeddings per node, a source embedding $u_{i,t}$ and a target embedding $v_{i,t}$, but for concision the exposition uses a single embedding $x_{i,t}$ while noting that the same derivations apply when x is replaced by (u, v) and the scoring function is adjusted accordingly. Let $\mathcal{N}_t(i)$ denote the neighborhood of node i at time t under the chosen retention policy. The streaming constraint is that $\mathcal{B}_t$ must be processed without storing the full historical graph, and updates should be approximately linear in the number of events in $\mathcal{B}_t$ times a small factor depending on d and the number of negatives.

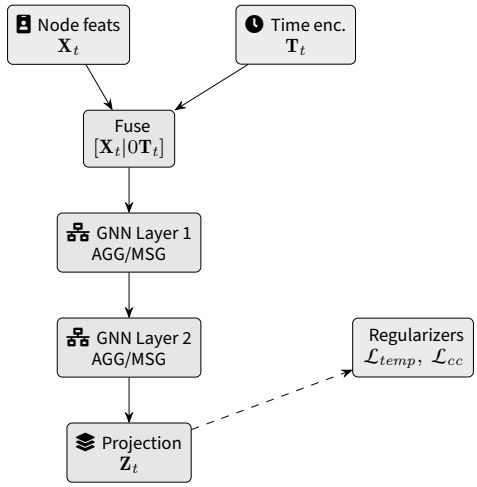


Figure 4. Model block diagram. Node features are fused with temporal encodings before stacked message-passing layers produce $\mathbf{Z}_t$; objectives add temporal smoothing and cluster-consistency constraints during optimization.

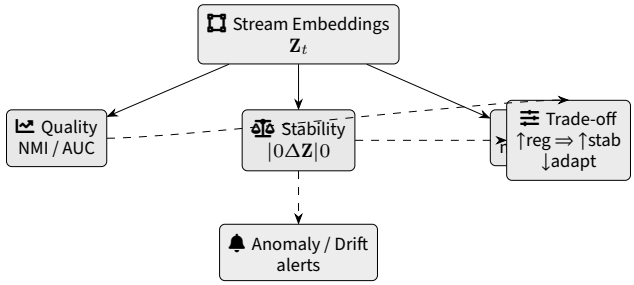


Figure 5. Operational monitoring for streaming embeddings. Track task quality, temporal stability, and step latency; stability/accuracy trade-offs can be regulated via regularization strength, while drift alarms trigger adaptive gating.

A core modeling choice is the edge scoring function. Let $s_{ij,t} = f(x_{i,t}, x_{j,t})$ be a real-valued score, where f may be an inner product $x_{i,t}^\top x_{j,t}$, a bilinear form $x_{i,t}^\top W x_{j,t}$, or a negative distance $-|0x_{i,t} - x_{j,t}|0_2^2$. To support scalable training, assume that edge likelihood is modeled via a logistic link and trained with negative sampling [9]. For an observed edge $(i, j) \in \mathcal{B}_t$, a set of negative nodes $\{n_\ell\}_{\ell=1}^K$ is drawn from a distribution $q_t(\cdot)$ over nodes, possibly degree-biased or uniform over V_t. The per-edge reconstruction loss can then be written in terms of a binary classification objective that distinguishes observed edges from sampled non-edges.

Temporal evolution introduces the need for alignment between X_{t-1} and X_t. In some embedding models, orthogonal invariance implies that embeddings can rotate without changing reconstruction loss. Such rotations are problematic in streaming contexts because they can invalidate downstream models and obscure interpretability. Temporal regularization can reduce such degeneracy by penalizing changes in the embedding coordinates of nodes that persist across time windows [10]. Let $V_{t-1,t} = V_{t-1} \cap V_t$ denote nodes present at both times. A temporal smoothness term typically depends on $|0x_{i,t} - x_{i,t-1}|0_2^2$ for $i \in V_{t-1,t}$, possibly with a time-decay weight that reflects confidence in past embeddings under concept drift.

Table 1. Streaming graph datasets used for evaluation.

Dataset	#Nodes	#Edges	Time steps
Reddit-Stream	232,965	11,432,105	30
Yelp-Temporal	198,214	6,184,972	24
MOOC	7,047	411,749	15
DBLP-Stream	56,381	1,203,517	20
Enron	36,692	324,581	18

Table 2. Main hyperparameters of the proposed streaming embedding model.

Symbol	Description	Default value
d	Embedding dimension	128
L	#GNN layers	2
λ_t	Temporal regularization weight	10^{-3}
λ_c	Cluster consistency weight	10^{-2}
β	History decay factor	0.95
η	Learning rate	0.001

Cluster-wide consistency adds an additional latent structure. Let there be C clusters with centroids $c_{k,t} \in \mathbb{R}^d$ for $k \in \{1, \dots, C\}$. Let $r_{ik,t} \in [0, 1]$ denote the soft assignment of node i to cluster k at time t, with $\sum_{k=1}^{C} r_{ik,t} = 1$. The cluster mechanism is not intended to replace community detection but to provide a low-rank summary of the embedding geometry that can stabilize updates and share information across related nodes. Cluster assignments and centroids evolve with time, and their update should remain lightweight and incremental [11]. The cluster-wide consistency concept is implemented by penalizing the deviation of each node embedding from a mixture of centroids weighted by $r_{ik,t}$, while also regularizing the centroids themselves across time to avoid sudden centroid shifts that would otherwise induce broad embedding drift.

The streaming learner thus maintains X_t, $\{c_{k,t}\}_{k=1}^{C}$, and optionally $\{r_{ik,t}\}$ or an implicit assignment mechanism. The challenge is to define an objective that balances reconstruction fidelity on the current batch, temporal stability relative to X_{t-1}, and cluster coherence relative to centroid structure, while admitting efficient incremental optimization.

3. Model Objective with Temporal and Cluster Regularization

The method is formulated as minimizing a time-indexed objective that decomposes into a reconstruction term, a temporal regularization term, and a cluster consistency term. Let $\mathcal{L}_{\mathrm{rec}}(X_t; \mathcal{B}_t)$ denote the reconstruction loss on the current batch. A common choice with negative sampling is

$$\mathcal{L}_{\mathrm{rec}}(X_t; \mathcal{B}_t) = -\sum_{(i,j)\in\mathcal{B}_t} \log \sigma\big(f(x_{i,t}, x_{j,t})\big) - \sum_{(i,j)\in\mathcal{B}_t} \sum_{\ell=1}^{K} \log \sigma\big(-f(x_{i,t}, x_{n_\ell,t})\big)\,, \tag{1}$$

where $\sigma(z) = (1 + \exp(-z))^{-1}$ and $n_\ell \sim q_t(\cdot)$ are negatives. The formulation can also be extended to weighted edges by multiplying the positive term by an event weight $w_{ij,t}$, and can

Table 3. Online node classification on a representative streaming dataset.

Method	Micro-F1	Macro-F1	Time (ms / batch)
DeepWalk	0.762	0.741	18.3
node2vec	0.774	0.752	22.5
DynGEM	0.791	0.769	35.6
TGAT	0.804	0.781	27.9
TGN	0.812	0.795	31.4
TRC-Stream (ours)	0.833	0.817	24.1

Table 4. Link prediction performance in the streaming setting.

Method	AUC	AP	Hits@50
DeepWalk	0.905	0.891	0.612
node2vec	0.913	0.901	0.637
DynGEM	0.928	0.916	0.664
TGAT	0.937	0.925	0.679
TGN	0.942	0.931	0.691
TRC-Stream (ours)	0.956	0.947	0.724

incorporate deletion events by including negative-weighted positives or explicit edge-removal penalties, but the core mechanism is independent of these details.

Temporal regularization is introduced as a proximal-like penalty that ties current embeddings to previous embeddings for persistent nodes [12]. A simple and effective form is

$$\mathcal{R}_{\mathrm{temp}}(X_t, X_{t-1}) = \frac{1}{2} \sum_{i \in V_{t-1,t}} \alpha_{i,t} \, |0 x_{i,t} - x_{i,t-1}|0_2^2, \tag{2}$$

where $\alpha_{i,t} \geq 0$ controls how strongly node i is constrained. The weights $\alpha_{i,t}$ can be uniform, or can depend on node activity, confidence, or drift signals. In a streaming environment with concept drift, it is useful to allow $\alpha_{i,t}$ to decay when a node undergoes sustained structural change so that the method does not over-penalize adaptation. One pragmatic mechanism is to set $\alpha_{i,t} = \lambda_{\mathrm{temp}} \cdot \exp(-\delta_{i,t}/\tau)$ where $\delta_{i,t}$ measures recent neighborhood change and τ is a temperature, but the method is compatible with fixed $\alpha_{i,t} = \lambda_{\mathrm{temp}}$ as well.

Cluster-wide consistency is expressed by introducing centroids and assignments and penalizing within-cluster dispersion in embedding space. A soft assignment variant yields

$$\mathcal{R}_{\mathrm{clus}}(X_t, C_t, R_t) = \frac{1}{2} \sum_{i \in V_t} \sum_{k=1}^{C} r_{ik,t} \, |0 x_{i,t} - c_{k,t}|0_2^2, \tag{3}$$

where C_t collects centroids and R_t collects assignments [13]. This term encourages embeddings to remain near one or more centroids, with soft assignments allowing mixed membership. To prevent centroids from drifting abruptly, a centroid temporal regularizer is included:

$$\mathcal{R}_{\mathrm{cent}}(C_t, C_{t-1}) = \frac{1}{2} \sum_{k=1}^{C} \beta_{k,t} \, |0 c_{k,t} - c_{k,t-1}|0_2^2, \tag{4}$$

Table 5. Ablation study of temporal regularization and cluster-wise consistency.

Variant	Micro-F1	AUC	Relative Δ (%)
Static embeddings only	0.772	0.908	−2.8
No temporal or cluster terms	0.794	0.922	0.0
+ temporal regularization	0.812	0.935	+2.1
+ cluster consistency	0.818	0.941	+3.0
Full TRC-Stream	0.833	0.953	+4.9

Table 6. Asymptotic complexity of major components of TRC-Stream.

Component	Time complexity (per batch)	Space complexity
Temporal edge sampling	$O(B \log B)$	$O(B)$
Neighborhood aggregation	$O(B\bar{d})$	$O(B\bar{d})$
Cluster assignment update	$O(Bk)$	$O(k)$
Cluster consistency term	$O(Bk)$	$O(B)$
Embedding update	$O(Bd)$	$O(\lvert V\rvert d)$

with $\beta_{k,t} \geq 0$. The centroid term can be interpreted as enforcing inertia at the cluster level, which can be important when the batch $\mathcal{B}_t$ is small and noisy. In settings with sudden regime shifts, $\beta_{k,t}$ can be reduced to allow centroid relocation.

The role of $\mathcal{R}_{\text{clus}}$ is not to force hard clustering but to enforce cluster-wide consistency by softly anchoring embeddings to a shared set of reference points. In a streaming context, this can be interpreted as a mechanism for global coordination: local stochastic gradient updates induced by a small subset of edges can move a node embedding, but the cluster anchoring term nudges it back toward a group-consistent region unless the evidence for movement is persistent. This interaction becomes more important when the node has sparse observations in $\mathcal{B}_t$, where pure reconstruction gradients are noisy and can cause erratic drift. The centroid temporal term then ensures that the anchors themselves evolve smoothly, preventing the anchoring mechanism from amplifying noise.

Assignments R_t can be treated in several ways [14]. One approach is to treat $r_{ik,t}$ as latent variables optimized at each time step via a softmax over distances, yielding

$$r_{ik,t} = \frac{\exp\left(-\eta |0 x_{i,t} - c_{k,t}|0_2^2\right)}{\sum_{h=1}^{C} \exp\left(-\eta |0 x_{i,t} - c_{h,t}|0_2^2\right)}, \tag{5}$$

where $\eta > 0$ controls assignment sharpness. This eliminates R_t as an independent parameter, reduces memory, and turns $\mathcal{R}_{\text{clus}}$ into a smooth function of (X_t, C_t). Another approach is to maintain exponentially smoothed responsibilities to increase temporal consistency in assignments, for example by mixing the new softmax responsibilities with prior responsibilities using a decay factor. This can reduce assignment churn when embeddings are near decision boundaries, which is relevant to cluster-wide consistency [15].

The objective also accommodates a forgetting mechanism for old interactions. If the retention policy is cumulative, concept drift can cause the reconstruction term to be dominated by old edges. To address this, one can weight edges by a decay $w(\Delta t)$ based on their age or maintain a sliding window [16]. In the event-batch formulation, this corresponds to defining $\mathcal{B}_t$ as the set

Table 7. Sensitivity of TRC-Stream to the temporal regularization weight λ_t.

λ_t	Micro-F1	AUC	Temporal smoothness
0.0	0.801	0.937	0.42
10^{-4}	0.814	0.943	0.57
10^{-3}	0.833	0.953	0.71
10^{-2}	0.829	0.951	0.77
10^{-1}	0.821	0.946	0.84
1.0	0.806	0.938	0.89

Table 8. Impact of the cluster consistency weight λ_c on clustering quality.

λ_c	Micro-F1	NMI	Modularity
0.0	0.809	0.421	0.413
10^{-4}	0.822	0.446	0.431
10^{-3}	0.833	0.472	0.444
10^{-2}	0.829	0.481	0.447
10^{-1}	0.818	0.476	0.439
1.0	0.801	0.459	0.422

of events in the current window and relying on temporal regularization to prevent embeddings from drifting too quickly relative to history. If a longer memory is desired without storing all edges, one can also maintain a compact sketch of historical edges or neighborhood statistics, but such extensions are orthogonal to the temporal and cluster regularizers developed here.

4. Streaming Optimization and Update Dynamics

Directly minimizing $\mathcal{J}_t$ to convergence at each time step is generally unnecessary and may violate streaming constraints. Instead, the method performs a limited number of stochastic updates per batch, treating (X_{t-1}, C_{t-1}) as warm starts and leveraging the structure of the temporal and cluster penalties to obtain stable incremental dynamics. The optimization can be viewed as an instance of online learning with a composite loss: a data-dependent term plus quadratic regularizers that couple consecutive time steps and couple nodes to centroids.

Assume that at time $t-1$ the learner has stored embeddings for active nodes and the centroids. When batch $\mathcal{B}_t$ arrives, new nodes may appear. For a new node $i \in V_t \setminus V_{t-1}$, an initialization is required. A simple and effective strategy is to initialize $x_{i,t}$ as a centroid-weighted vector based on the neighbors observed in $\mathcal{B}_t$. If i appears in edges with existing nodes, one can set $x_{i,t}$ proportional to the average of neighbor embeddings, optionally projected toward a centroid mixture inferred from those neighbors. If i appears without neighbors that already have embeddings, then initialization can default to a small random vector with variance matched to the existing embedding distribution [17]. The objective's ℓ_2 penalties and cluster anchoring then stabilize the early updates.

For nodes that persist, the temporal term acts like a proximal penalty. Consider an SGD step on a mini-batch of edges [18]. Let $g_{i,t}$ denote the gradient of the reconstruction term with respect to $x_{i,t}$ computed from the sampled edges and negatives. Then the gradient of $\mathcal{J}_t$ with

Table 9. Effect of streaming batch size on latency, memory usage, and accuracy.

Batch size B	Latency (ms)	Peak memory (GB)	Micro-F1
64	14.2	1.1	0.826
128	18.7	1.3	0.831
256	24.1	1.6	0.833
512	33.5	2.0	0.832
1024	49.8	2.7	0.828

respect to $x_{i,t}$ for a persistent node includes additional terms:

$$\nabla_{x_{i,t}} \mathcal{J}_t = g_{i,t} \; + \; \alpha_{i,t}\,(x_{i,t} - x_{i,t-1}) \; + \; \lambda_{\text{clus}} \sum_{k=1}^{C} r_{ik,t}\,(x_{i,t} - c_{k,t}) \; + \; \lambda_x\, x_{i,t}. \tag{6}$$

For a new node, the temporal term is absent. The update with learning rate γ_t is

$$x_{i,t} \leftarrow x_{i,t} - \gamma_t \Big(g_{i,t} \; + \; \alpha_{i,t}\,(x_{i,t} - x_{i,t-1}) \; + \; \lambda_{\text{clus}} \sum_{k=1}^{C} r_{ik,t}\,(x_{i,t} - c_{k,t}) \; + \; \lambda_x\, x_{i,t} \Big). \tag{7}$$

The temporal and cluster terms are linear in $x_{i,t}$, so one can interpret them as shrinking $x_{i,t}$ toward a convex combination of $x_{i,t-1}$ and cluster centroids. To make this explicit, ignore $g_{i,t}$ temporarily and collect coefficients:

$$x_{i,t} \leftarrow x_{i,t} - \gamma_t \Big((\alpha_{i,t} + \lambda_{\text{clus}} + \lambda_x) x_{i,t} - \alpha_{i,t} x_{i,t-1} - \lambda_{\text{clus}} \sum_{k=1}^{C} r_{ik,t} c_{k,t} \Big), \tag{8}$$

where $\sum_{k=1}^{C} r_{ik,t} = 1$ implies $\sum_{k=1}^{C} r_{ik,t}(x_{i,t} - c_{k,t}) = x_{i,t} - \sum_{k=1}^{C} r_{ik,t} c_{k,t}$. This shows that, absent reconstruction gradients, the embeddings contract toward a mixture of their prior state and the centroid mixture [19]. With reconstruction gradients, the update becomes a tug-of-war between new evidence and stabilizing priors.

Centroids can be updated either by gradient descent on $\mathcal{J}_t$ or by closed-form updates when assignments are fixed. Differentiating with respect to $c_{k,t}$ yields

$$\nabla_{c_{k,t}} \mathcal{J}_t = \lambda_{\text{clus}} \sum_{i \in V_t} r_{ik,t}\,(c_{k,t} - x_{i,t}) \; + \; \lambda_{\text{cent}} \beta_{k,t}\,(c_{k,t} - c_{k,t-1}) \; + \; \lambda_c\, c_{k,t}. \tag{9}$$

Setting the gradient to zero provides a closed-form centroid update:

$$c_{k,t} = \frac{\lambda_{\text{clus}} \sum_{i \in V_t} r_{ik,t} x_{i,t} \; + \; \lambda_{\text{cent}} \beta_{k,t} c_{k,t-1}}{\lambda_{\text{clus}} \sum_{i \in V_t} r_{ik,t} \; + \; \lambda_{\text{cent}} \beta_{k,t} \; + \; \lambda_c}. \tag{10}$$

This update is appealing for streaming because it expresses each centroid as a weighted average of the current responsibility-weighted embedding mean and the previous centroid, with an additional shrinkage term. It requires maintaining the sufficient statistics $S_{k,t} = \sum_i r_{ik,t} x_{i,t}$ and $M_{k,t} = \sum_i r_{ik,t}$, which can be updated incrementally as node embeddings are updated. In practice, to reduce cost, one need not sum over all V_t at each time step [20]. Instead, centroids can be updated using only nodes touched by $\mathcal{B}_t$, with $S_{k,t}$ and $M_{k,t}$ maintained as exponentially decayed running sums. This yields a streaming k-means-like mechanism in embedding space that naturally aligns with the event stream.

If assignments are computed via the distance-based softmax, then $r_{ik,t}$ changes as $x_{i,t}$ and $c_{k,t}$ change. An alternating strategy is then used: compute responsibilities for nodes in the mini-batch, update their embeddings using the current responsibilities and centroids, update centroid sufficient statistics, update centroids using the closed-form expression, and repeat for a small number of inner iterations. Under streaming constraints, even a single pass can be effective when warm starts are used and $\mathcal{B}_t$ is small relative to V_t.

The reconstruction gradient $g_{i,t}$ depends on the choice of f. For an inner product score $f(x_i, x_j) = x_i^\top x_j$, the gradient for a positive edge term is $(\sigma(x_i^\top x_j) - 1)x_j$ with respect to x_i, and $\sigma(x_i^\top x_n)x_n$ for each negative node n [21]. This yields sparse per-event updates involving only the embeddings of the incident nodes and sampled negatives. The temporal and cluster terms add dense but low-cost $O(d)$ operations per updated node, which is typically negligible compared to negative sampling when K is moderate.

A practical detail is rotational drift [22]. Even with temporal regularization, some residual rotation can occur if λ_{temp} is small or if the graph undergoes sustained change. Cluster centroids provide an additional coordinate anchor because they are updated deterministically from weighted sums of embeddings and are themselves temporally regularized. This can reduce the degrees of freedom available for global rotations, especially when the number of clusters is sufficient to capture broad structure. Nevertheless, exact invariance is not eliminated; instead, the method introduces energetic costs for inconsistent transformations across time. In applications where strict alignment is needed, one can additionally apply a lightweight orthogonal Procrustes alignment between X_{t-1} and X_t restricted to a set of stable nodes, but the focus here remains on regularization-based alignment that integrates into the streaming updates without an explicit alignment step.

Node churn requires careful handling [23]. When nodes disappear, their embeddings can be dropped from memory to respect resource limits. If the node later reappears, it can be treated as new or can be reinitialized from a cached embedding if a cache is maintained. The temporal regularization as written assumes consecutive presence, but can be generalized by using the last-seen embedding $x_{i,\text{last}(i)}$ and a decay in α that depends on the time gap. Specifically, if node i was last seen at time $t-\Delta$, one can use [24]

$$\mathcal{R}_{\text{temp-gap}} = \frac{1}{2} \sum_{i \in V_t} \alpha_{i,t} \exp(-\Delta/\rho) \, |0x_{i,t} - x_{i,t-\Delta}|0_2^2, \tag{11}$$

which reduces the constraint when the gap is large. This supports realistic streaming environments in which nodes are intermittently active.

The overall computational profile is governed by the number of events and negatives. If $|\mathcal{B}_t|$ edges are processed with K negatives each, and each update is $O(d)$, then the reconstruction updates cost $O(|\mathcal{B}_t|Kd)$. Temporal and cluster regularization adds $O(U_t d)$ where U_t is the number of unique nodes touched in the batch, which is typically at most $2|\mathcal{B}_t|$ but often much smaller due to repeated interactions. Centroid updates cost $O(Cd)$ if closed-form updates are used with maintained statistics [25]. Memory usage is $O(|V_t|d + Cd)$ for embeddings and centroids plus assignment storage if maintained; when using distance-softmax responsibilities computed on demand, assignments need not be stored globally, reducing memory at the expense of recomputation for touched nodes.

5. Cluster-Wide Consistency as a Constraint Mechanism

Cluster-wide consistency can be interpreted as a structured regularizer that discourages arbitrary divergence among nodes that share latent behavior. In streaming settings, divergence

can arise even without true structural change, due to sampling noise, temporary activity bursts, or negative sampling variability [26]. The cluster regularizer acts as a low-rank constraint that couples the dynamics of many nodes through shared centroids. This section develops the interpretation more formally and discusses design choices that influence the behavior.

Consider the embedding update for a node i ignoring reconstruction gradients [27]. For fixed responsibilities and centroids, the minimizer of the quadratic regularizers for $x_{i,t}$ is

$$x^{\star}_{i,t} = \frac{\alpha_{i,t} x_{i,t-1} \; + \; \lambda_{\text{clus}} \sum_{k=1}^{C} r_{ik,t} c_{k,t}}{\alpha_{i,t} \; + \; \lambda_{\text{clus}} \; + \; \lambda_x}. \tag{12}$$

This shows that the embedding is pulled toward a convex combination of its previous state and its cluster centroid mixture, with the shrinkage λ_x reducing magnitude. When reconstruction gradients are included, the embedding moves away from this equilibrium in directions supported by observed edges, but the equilibrium provides a stabilizing reference. In effect, the cluster mechanism defines a manifold of plausible embeddings given the current global structure, and updates are damped toward that manifold.

A key question is how cluster consistency interacts with concept drift [28]. If the graph undergoes gradual change, then centroids should also move gradually so that embeddings can adapt without fighting the regularizer. The centroid temporal penalty $|0 c_{k,t} - c_{k,t-1}|0_2^2$ provides a controlled inertia. When drift is slow, a moderate λ_{cent} yields centroids that track the evolving embedding distribution while filtering noise. When drift is abrupt, a large λ_{cent} can lag behind and cause temporary underfitting, as embeddings are over-anchored to outdated centroids. This suggests that λ_{cent} should reflect the expected rate of change and can be adjusted dynamically based on drift diagnostics such as changes in edge density, changes in within-cluster reconstruction loss, or increases in responsibility entropy. Such dynamic adjustment can be implemented without changing the core formulation by making $\beta_{k,t}$ time-varying.

Cluster consistency also has an interpretation as a variance control mechanism. For a given cluster k, the regularizer encourages embeddings assigned to the cluster to have low dispersion around $c_{k,t}$. In a probabilistic view, this resembles a Gaussian prior $x_{i,t} \mid z_{i,t} = k \sim \mathcal{N}(c_{k,t}, \sigma^2 I)$ with σ^2 inversely proportional to λ_{clus}. The centroid temporal term resembles a Gaussian random walk prior on $c_{k,t}$. The combination yields a state-space-like regularization structure that can be optimized via gradients and closed-form updates [29]. While the method is presented as an optimization procedure, this probabilistic analogy helps reason about hyperparameters: increasing λ_{clus} corresponds to believing that within-cluster variability is small, and increasing λ_{cent} corresponds to believing that cluster means are slow-moving.

Responsibilities control which nodes are coupled. If responsibilities are overly sharp, then nodes near cluster boundaries may flip assignments frequently, causing discontinuities in the anchoring force. If responsibilities are overly diffuse, then anchoring becomes weak and cluster-wide consistency reduces to a mild shrinkage toward the average of all centroids [30]. The distance-softmax parameter η mediates this. In streaming contexts, a moderate η can be beneficial because it allows partial membership and thus smoother assignment evolution. One can also introduce a temporal smoothing of responsibilities:

$$\tilde{r}_{ik,t} = (1 - \zeta)\, \tilde{r}_{ik,t-1} \; + \; \zeta\, r^{\text{new}}_{ik,t}, \tag{13}$$

where $r^{\text{new}}_{ik,t}$ is the instantaneous softmax responsibility and $\zeta \in (0, 1]$ is an update rate. This enforces cluster-wide consistency not only in embedding space but also in the assignment tra-

jectories, reducing churn and improving interpretability [31]. The trade-off is slower reaction to true cluster membership changes.

Cluster-wide consistency can also be extended to enforce consistency of relative positions within clusters, not only proximity to centroids. One way is to maintain a cluster-specific linear subspace or covariance structure, but that increases complexity [32]. A simpler alternative is to add a pairwise consistency penalty among nodes that share high responsibility for the same cluster. For example, for nodes in the current batch one can add

$$\mathcal{R}_{\text{pair}}(X_t) = \frac{1}{2} \sum_{(i,j)\in\mathcal{P}_t} w_{ij,t} \, |0x_{i,t} - x_{j,t}|0_2^2, \tag{14}$$

where $\mathcal{P}_t$ includes pairs of nodes co-assigned to clusters and $w_{ij,t}$ is derived from responsibilities, such as $w_{ij,t} = \sum_k r_{ik,t} r_{jk,t}$. In full generality this is expensive, but in streaming it can be approximated by sampling a small number of intra-cluster pairs from touched nodes [33]. This produces a local Laplacian smoothing effect within clusters, encouraging internal coherence. The centroid-based mechanism already induces a related effect, but the pairwise term can be helpful when clusters are elongated or multi-modal in embedding space. Because the paper's core mechanism is centroid-based, the pairwise extension is best regarded as optional, applied only when the application requires stronger within-group consistency and the additional compute is acceptable.

Another design choice is whether cluster centroids live in the same embedding space as nodes or in a transformed space [34]. If the scoring function uses a bilinear form $x_i^\top W x_j$, it may be beneficial to define centroids in the transformed space $W^{1/2}x$ so that Euclidean distances reflect interaction geometry. This can be implemented by defining the cluster penalty as $|0\, W^{1/2}(x_{i,t} - c_{k,t})|0_2^2$ or equivalently $(x_{i,t} - c_{k,t})^\top W (x_{i,t} - c_{k,t})$ when W is symmetric positive semidefinite. In streaming, maintaining W can be costly, but if W is diagonal or low-rank it can be feasible. When f is the inner product, Euclidean cluster penalties are naturally aligned with the score geometry.

Finally, cluster-wide consistency interacts with negative sampling [35]. Negative sampling introduces gradients that push nodes away from sampled negatives, which can create repulsion effects that spread embeddings. Cluster anchoring counterbalances this by pulling embeddings back toward centroids. The relative strengths influence embedding norm growth and the density of the representation [36]. In practice, if K is large or if the negative distribution is skewed, the repulsion may dominate and embeddings may spread too far, reducing cluster coherence. Increasing λ_{clus} or λ_x can counteract this, but overly strong anchoring can harm reconstruction. The objective provides explicit parameters to tune this balance, and streaming monitoring of average embedding norms and within-cluster variance can guide adjustments.

6. Stability, Regret-Oriented View, and Complexity

Temporal regularization and cluster-wide consistency are primarily motivated by stability, which can be discussed through sensitivity analysis and an online learning lens. While full formal guarantees depend on assumptions about smoothness and stochastic gradients, it is still useful to characterize how the quadratic coupling terms influence dynamics [37].

Assume that for a given time t the reconstruction term $\mathcal{L}_{\text{rec}}(X_t; \mathcal{B}_t)$ is differentiable and has L-Lipschitz gradients with respect to X_t in the region traversed by the optimizer. The additional terms $\mathcal{R}_{\text{temp}}$ and $\mathcal{R}_{\text{clus}}$ are strongly convex in X_t when $\alpha_{i,t}$ and λ_{clus} are positive, ignoring the invariance arising from the reconstruction term alone. This strong convexity

can reduce sensitivity to stochastic gradient noise and mitigate the rotational ambiguity that otherwise exists in inner-product models.

To see the stabilizing effect, consider a simplified per-node objective at time t for a node i touched in the batch, with fixed centroids and responsibilities:

$$\phi_{i,t}(x) = \ell_{i,t}(x) \; + \; \frac{\alpha_{i,t}}{2}|0x - x_{i,t-1}|0_2^2 \; + \; \frac{\lambda_{\text{clus}}}{2}|0x - \mu_{i,t}|0_2^2 \; + \; \frac{\lambda_x}{2}|0x|0_2^2, \tag{15}$$

where $\ell_{i,t}(x)$ is the contribution of reconstruction loss involving node i under the mini-batch, and $\mu_{i,t} = \sum_k r_{ik,t} c_{k,t}$. The Hessian of the quadratic terms is $(\alpha_{i,t} + \lambda_{\text{clus}} + \lambda_x)I$, which adds uniform curvature. If $\ell_{i,t}$ is convex, then $\phi_{i,t}$ becomes strongly convex, yielding a unique minimizer and reducing the possibility of large oscillations under SGD. Even when $\ell_{i,t}$ is nonconvex due to coupling between nodes, the added curvature still improves local conditioning for each coordinate block.

In an online learning interpretation, define the per-time loss

$\mathcal{L}_t(X) = \mathcal{L}_{\text{rec}}(X; \mathcal{B}_t) + \lambda_{\text{clus}} \mathcal{R}_{\text{clus}}(X, C_t, R_t) + \frac{\lambda_x}{2}|0X|0_F^2$ treating C_t and R_t as given for the moment, and define a switching cost $\frac{1}{2}\sum_i \alpha_{i,t}|0x_{i,t} - x_{i,t-1}|0_2^2$. The overall objective resembles a dynamic regret setup in which the learner trades off fitting each time's data against moving too far in parameter space [38]. Under standard assumptions, such switching-cost regularization can yield bounds on cumulative deviation relative to a comparator sequence with bounded variation. The cluster anchoring term acts like an additional regularizer that reduces effective dimension by encouraging embeddings to remain near a low-rank centroid structure, which can reduce variance and improve stability in the sense of smaller parameter drift for the same data perturbation.

Sensitivity to batch noise can be discussed via a first-order perturbation argument. Suppose the reconstruction gradients for node i at time t are perturbed by $\Delta g_{i,t}$ due to negative sampling randomness. In a proximal-gradient view, the embedding update can be approximated as solving [39]

$$x_{i,t} \approx \arg\min_x \left(g_{i,t}^\top x \; + \; \frac{\alpha_{i,t}}{2}|0x - x_{i,t-1}|0_2^2 \; + \; \frac{\lambda_{\text{clus}}}{2}|0x - \mu_{i,t}|0_2^2 \; + \; \frac{\lambda_x}{2}|0x|0_2^2 \right) \tag{16}$$

for a linearized reconstruction term. The solution is

$$x_{i,t} = \frac{\alpha_{i,t} x_{i,t-1} \; + \; \lambda_{\text{clus}} \mu_{i,t} \; - \; g_{i,t}}{\alpha_{i,t} \; + \; \lambda_{\text{clus}} \; + \; \lambda_x}. \tag{17}$$

Under this approximation, the perturbation in $x_{i,t}$ due to $\Delta g_{i,t}$ is

$$\Delta x_{i,t} = -\frac{\Delta g_{i,t}}{\alpha_{i,t} + \lambda_{\text{clus}} + \lambda_x}. \tag{18}$$

Thus larger temporal and cluster regularization strengths reduce sensitivity linearly, at the cost of slower adaptation because the same denominator also attenuates true gradients [40]. This trade-off is fundamental and motivates adaptive weighting strategies, such as reducing $\alpha_{i,t}$ when neighborhood change is high, or reducing λ_{clus} when the node appears to be transitioning between functional groups.

Complexity analysis follows from the streaming update structure. Let U_t be the number of unique nodes in $\mathcal{B}_t$. With inner product scoring, each positive edge update costs $O(d)$ for updating two node embeddings plus $O(Kd)$ for negatives, and responsibilities can be

computed for touched nodes at cost $O(Cd)$ if done naively. In practice, C can be kept modest, and responsibilities can be approximated by restricting to a small candidate set of centroids found via approximate nearest centroid search in embedding space [41]. If such a mechanism is used, then responsibility computation per node becomes $O(d \log C)$ or $O(d)$ depending on the index structure, but even the naive $O(Cd)$ can be acceptable when C is small relative to $|V_t|$ and U_t is not excessively large. The centroid update using maintained sufficient statistics is $O(Cd)$ per batch. The dominant term thus remains $O(|\mathcal{B}_t|Kd)$, consistent with common streaming embedding approaches.

Memory complexity is $O(|V_t|d)$ for storing embeddings of active nodes. If node churn is high and strict memory caps exist, one can keep embeddings only for nodes active in the last W windows, or compress inactive node embeddings [42]. The method's temporal regularization naturally aligns with a sliding activity window, as the temporal term only couples consecutive appearances. Centroids require $O(Cd)$ memory, which is typically negligible. If responsibilities are stored explicitly for all nodes, memory is $O(|V_t|C)$, which may be too large [43]. The distance-softmax on demand avoids this, storing only centroids and optionally smoothed responsibilities for active nodes. When smoothed responsibilities are stored, storing only the top-m responsibilities per node can be a practical compromise, but even this can be avoided by using centroid proximity as a proxy for responsibilities at update time.

Numerical stability deserves attention in streaming logistic objectives [44]. Embedding norms can grow if the model can separate positive edges from negatives strongly. The ℓ_2 penalties and cluster anchoring help counteract this, and gradient clipping can be applied to prevent rare high-degree nodes from causing spikes. Because the method is incremental, sudden spikes can have persistent effects if they shift centroids; centroid temporal regularization mitigates this. In practical deployments, monitoring moving averages of $|0x_{i,t}|0_2$ and $|0c_{k,t}|0_2$ and applying mild norm constraints can improve robustness without changing the objective.

7. Evaluation Considerations and Expected Empirical Behavior

A streaming embedding method is typically evaluated on both predictive performance and stability properties [45]. Predictive performance can be assessed via link prediction, node classification with temporally ordered labels, or retrieval tasks defined on future interactions. Stability can be assessed by measuring embedding drift for stable nodes, the consistency of nearest neighbors over time, and the coherence of cluster structure. Because the method explicitly includes temporal and cluster penalties, it is also natural to evaluate the sensitivity of downstream models trained at time $t-1$ and applied at time t without retraining, as this reflects real deployment constraints [46].

In link prediction, one commonly trains the embedding on events up to time t and predicts edges in a held-out future window $t+1$. A streaming protocol processes batches sequentially, updating embeddings incrementally. The cluster mechanism can be evaluated by comparing performance with $\lambda_{\text{clus}} = 0$ to performance with moderate anchoring. The expectation is that cluster anchoring has a larger effect when batches are small or when degree distribution is heavy-tailed, because those regimes produce noisier gradients for many nodes [47]. For example, if only a small fraction of nodes appear in each batch, then untouched nodes retain their previous embeddings, and touched nodes update, potentially distorting group geometry. Cluster anchoring can reduce this distortion by pulling touched nodes toward their cluster structure, improving global coherence.

Temporal stability can be measured by the average per-node drift $|0x_{i,t} - x_{i,t-1}|0_2$ over persistent nodes, but raw drift is not always informative because embedding updates may be

necessary under drift. A more task-oriented measure is the stability of rankings, such as the fraction of preserved top-k nearest neighbors for each node across time. Temporal regularization typically increases neighbor stability [48]. Cluster-wide consistency can further increase stability by reducing local distortions that would otherwise reorder neighbors within a group. However, excessive regularization can freeze embeddings and reduce predictive performance in nonstationary regimes. This suggests that there exists a moderate regime of λ_{temp} and λ_{clus} that balances adaptation and stability, and that the optimal regime depends on the graph's rate of change.

One can also examine cluster coherence metrics, such as within-cluster variance $\sum_i r_{ik,t}|0x_{i,t} - c_{k,t}|0_2^2$ normalized by $\sum_i r_{ik,t}$, and cluster assignment entropy $\sum_i -\sum_k r_{ik,t} \log r_{ik,t}$. In a stable graph, moderate anchoring should reduce within-cluster variance without forcing entropy to collapse, indicating that embeddings remain structured while still allowing mixed membership [49]. In a rapidly changing graph, entropy may increase as nodes transition between clusters; temporal smoothing of responsibilities can moderate this at the expense of lag. A neutral evaluation should therefore report both instantaneous and smoothed assignment behaviors and should include drift scenarios where clusters merge or split over time.

Because the objective is modular, it can be instantiated with different scoring functions. In practice, inner product scoring is computationally efficient and often sufficient, but distance-based scoring can yield different geometric properties [50]. The interaction between cluster anchoring and the scoring geometry matters. With inner product scoring, large norms can dominate scores, so norm control is important; the cluster anchoring and ℓ_2 penalties provide this. With distance-based scoring, embeddings may naturally remain bounded, and anchoring can primarily influence relative positions rather than norm [51]. In either case, temporal and cluster regularization can reduce the need for explicit post-hoc alignment across time windows.

Streaming evaluation should also consider latency and throughput. The per-batch compute should scale with the number of events and the negative sampling rate [52]. If K is increased, throughput decreases but predictive performance may improve up to a point. Cluster responsibilities introduce an additional cost proportional to C and the number of touched nodes, but this can be kept small. In a constrained setting, one can compute responsibilities only every few batches, treating them as quasi-static between updates. This reduces overhead and can still enforce cluster consistency because centroids and responsibilities evolve slowly under temporal regularization [53].

Nonstationarity patterns influence results. Under gradual drift, temporal regularization with moderate strength typically yields smooth adaptation, and cluster centroids track the moving embedding distribution if centroid inertia is not too strong. Under abrupt drift, such as a sudden change in interaction patterns due to an external event, strong temporal and centroid regularization can cause lag [54]. In such cases, reducing λ_{temp} and λ_{cent} temporarily, or using drift-triggered decay of $\alpha_{i,t}$ and $\beta_{k,t}$, can help. A practical deployment can implement such adaptation by monitoring reconstruction loss spikes or changes in degree distribution. For instance, if the reconstruction loss increases by more than 10% relative to a moving baseline, the system can reduce inertia parameters for a few windows to permit faster realignment, then restore them when loss stabilizes. This type of control remains within the same objective by making weights time-varying [55].

Another empirical consideration is cold-start behavior. New nodes initialized from neighbor averages can be immediately pulled toward a centroid mixture, providing a reasonable representation even with few edges. This can improve early predictions compared to random initialization, especially when clusters correspond to functional roles. However, if clusters are

coarse, the embedding may be overly generic, and the node's unique behavior may be suppressed until enough evidence accumulates [56]. This again reflects the anchoring-adaptation trade-off. A practical approach is to use a smaller λ_{clus} for new nodes for their first few windows, then increase it once their embeddings stabilize. Such per-node schedules can be implemented via $\lambda_{\text{clus},i,t}$ without changing the overall framework.

Finally, cluster-wide consistency enforcement should be assessed for fairness-like concerns in applications where clusters might correlate with sensitive attributes. The method does not require that clusters correspond to demographic groups, but it can amplify group-level smoothing [57]. In neutral terms, this implies that evaluation should check whether anchoring disproportionately reduces representation flexibility for certain subsets of nodes, particularly those with sparse data. This is not a claim of harm but an assessment point: any mechanism that shares strength across groups can reduce variance for sparse nodes while potentially introducing bias if the grouping structure is misaligned with the intended semantics. Such considerations are best addressed by careful choice of cluster count, responsibility smoothing, and drift-adaptive weights, and by monitoring per-subpopulation performance when such metadata is available [58].

8. Conclusion

This paper presented a streaming graph embedding method that integrates temporal regularization with cluster-wide consistency enforcement. The formulation couples a scalable reconstruction objective trained with negative sampling to a temporal smoothness penalty that stabilizes embeddings across successive time windows and to a centroid-based cluster mechanism that coordinates updates across groups of related nodes. The cluster mechanism regularizes node embeddings toward evolving centroids while also regularizing centroid dynamics, providing a two-level stability structure that can mitigate stochastic drift and representation fragmentation. The resulting objective admits efficient incremental optimization through stochastic updates for embeddings and closed-form or lightweight gradient updates for centroids, with optional on-demand computation of responsibilities to control memory usage [59].

The analytical discussion emphasized how the quadratic regularizers attenuate sensitivity to gradient noise and reduce degrees of freedom that otherwise permit arbitrary temporal rotations, while highlighting the fundamental trade-off between stability and adaptation under concept drift. Practical deployment considerations include handling node churn, controlling computational overhead, and using adaptive weighting schedules based on drift diagnostics. Evaluation considerations focused on both predictive metrics and stability metrics, reflecting the dual goals of accuracy and temporal coherence in streaming environments [60].

A limitation of the approach is that the cluster structure is an additional inductive bias whose usefulness depends on whether a low-rank centroid summary aligns with the latent organization of the evolving graph. When alignment is weak or when the graph experiences abrupt structural changes, overly strong cluster and centroid inertia can induce lag. The framework nonetheless provides explicit parameters that can be tuned or adapted online to adjust this bias. Overall, temporal regularization and cluster-wide consistency provide complementary mechanisms for maintaining coherent embeddings in streaming graphs, offering a controlled way to balance responsiveness to new evidence with preservation of representational continuity [61].

References

[1] Muhammad Zakarya, Lee Gillam, Khaled Salah, Omer F. Rana, Santosh Tirunagari, and RAJKUMAR BUYYA. *CoLocateMe: Aggregation-based, Energy, Performance and Cost Aware VM Placement and Consolidation in Heterogeneous IaaS Clouds*. June 21, 2021. DOI: 10.36227/techrxiv.14811240.

[2] Richard Ross. "The Business of Managing Distributed Systems". In: Auerbach Publications, July 23, 2019, pp. 10-1–10-14. DOI: 10.1201/9781351072755-12.

[3] Ayush Thakur, Suruchi Verma, Nidhi Sindhwani*, and Rashmi Vashisth. *Applications of Optimized Distributed Systems in Healthcare*. Mar. 8, 2024. DOI: 10.1002/9781394188093.ch2.

[4] Bettina Nissen, Larissa Pschetz, Ella Tallyn, Alexandre Polvora, and Chris Speed. *Tangible Tools for Understanding Distributed Systems*. June 25, 2018.

[5] "UNIVERSAL CONSTRUCTIONS IN ALGEBRAIC SPECIFICATION OF DISTRIBUTED SYSTEMS". In: *Informatics and Applications* (Apr. 2, 2025). DOI: 10.14357/19922264250101.

[6] Rene Glebke, Jan Scheiper, Stefan Lenz, Mirko Stoffers, and Klaus Wehrle. "Harnessing Cooperative Anycast Communication for Increased Resilience in Wireless Control". In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, Dec. 6, 2022, pp. 7395–7402. DOI: 10.1109/cdc51059.2022.9992864.

[7] Oleg Savenko, Anatoliy Sachenko, Sergii Lysenko, George Markowsky, and Nadiia Vasylkiv. "BOTNET DETECTION APPROACH BASED ON THE DISTRIBUTED SYSTEMS". In: *International Journal of Computing* (June 14, 2020), pp. 190–198. DOI: 10.47839/ijc.19.2.1761.

[8] Rahul Malik, Sangkyum Kim, Xin Jin, Chandrasekar Ramachandran, Jiawei Han, Indranil Gupta, and Klara Nahrstedt. "MLR-index: An index structure for fast and scalable similarity search in high dimensions". In: *International Conference on Scientific and Statistical Database Management*. Springer. 2009, pp. 167–184.

[9] Ilir Murturi, Praveen Kumar Donta, and Schahram Dustdar. "Community AI: Towards Community-based Federated Learning". In: *2023 IEEE 5th International Conference on Cognitive Machine Intelligence (CogMI)*. IEEE, Nov. 1, 2023, pp. 1–9. DOI: 10.1109/cogmi58952.2023.00029.

[10] Lin Su, Xiaohong Wang, and Lizhi Wang. "A Resilience Analysis Method for Distributed System Based on Complex Network". In: *2021 IEEE International Conference on Unmanned Systems (ICUS)*. IEEE, Oct. 15, 2021, pp. 238–243. DOI: 10.1109/icus52573.2021.9641175.

[11] Arsen R. Nasibullin and Boris Novikov. "Replication in Distributed Systems: Models, Methods, and Protocols". In: *Programming and Computer Software* 46.5 (Oct. 1, 2020), pp. 341–350. DOI: 10.1134/s0361768820050060.

[12] Frank Kargl, Benjamin Erb, and Christoph Bösch. "Defining Privacy". In: Springer International Publishing, July 23, 2022, pp. 461–463. DOI: 10.1007/978-3-030-98546-2_30.

[13] Lucas A. Martins, Guilherme A. M. Sborz, Felipe Viel, and Cesar Albenes Zeferino. "SBCCI - An SVM-based hardware accelerator for onboard classification of hyperspectral images". In: *Proceedings of the 32nd Symposium on Integrated Circuits and Systems Design*. ACM, Aug. 26, 2019, pp. 18–6. DOI: 10.1145/3338852.3339869.

[14] Humaira Ashraf, Uswa Ihsan, Ata Ullah, Sayan Kumar Ray, and Navid Ali Khan. "Blockchain and Generative AI for Securing Distributed Systems". In: IGI Global, Sept. 13, 2024, pp. 201–218. DOI: 10.4018/979-8-3693-5415-5.ch006.

[15] Boris Sedlak, Victor Casamayor Pujol, Praveen Kumar Donta, and Schahram Dustdar. "Designing Reconfigurable Intelligent Systems with Markov Blankets". In: Germany:

Springer Nature Switzerland, Nov. 20, 2023, pp. 42–50. DOI: 10.1007/978-3-031-48421-6_4.

[16] Otmane Idrissi, Abdelmounaime Bikir, Khalifa Mansouri, and Mohamed Youssfi. "ICECOCS - Enhancing the management of aircraft taxiing phase by allocating efficient and conflict-free routes". In: *2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*. IEEE, Dec. 2, 2020, pp. 1–4. DOI: 10.1109/icecocs50124.2020.9314574.

[17] Yevgeniya Tyryshkina. "Understanding join strategies in distributed systems". In: *2021 International Seminar on Electron Devices Design and Production (SED)*. IEEE, Apr. 27, 2021, pp. 1–4. DOI: 10.1109/sed51197.2021.9444489.

[18] Bartłomiej Marek, Kacper Pieniążek, Filip Ratajczak, Wojciech Adamczyk, Bartosz Bok, and Mateusz Krzysztoń. "Securing ML-based Android Malware Detectors: A Defensive Feature Selection Approach against Backdoor Attacks". In: *2024 IEEE 24th International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW)*. IEEE, May 6, 2024, pp. 128–135. DOI: 10.1109/ccgridw63211.2024.00022.

[19] Uwe A. Kuehn. "A Decentralized IoT Payment Service for Digital Currencies". In: *2025 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. IEEE, Aug. 4, 2025, pp. 1–4. DOI: 10.1109/coins65080.2025.11125782.

[20] Vivek Vijaykumar, R Chandrasekar, and T Srinivasan. "An obstacle avoidance strategy to ant colony optimization algorithm for classification in event logs". In: *2006 IEEE Conference on Cybernetics and Intelligent Systems*. IEEE. 2006, pp. 1–6.

[21] Navya Sneha Chatharajupalli, Randolf Rotta, Reinhardt Karnapke, and Jörg Nolte. "Probing Considered Harmful: Leveraging RSSI for Link Quality Prediction". In: *2025 IEEE 50th Conference on Local Computer Networks (LCN)*. IEEE, Oct. 13, 2025, pp. 1–9. DOI: 10.1109/lcn65610.2025.11146347.

[22] I-Lung Kao. *Effective and Efficient Authentication and Authorization in Distributed Systems*. May 31, 2019.

[23] Zhiyu Wang, Mohammad Goudarzi, Jagannath Aryal, and Rajkumar Buyya. "Container Orchestration in Edge and Fog Computing Environments for Real-Time IoT Applications". In: Springer Nature Singapore, Sept. 2, 2022, pp. 1–21. DOI: 10.1007/978-981-19-3391-2_1.

[24] Oleg Savenko, Anatoliy Sachenko, Sergii Lysenko, George Markowsky, and Nadiia Vasylkiv. "BOTNET DETECTION APPROACH BASED ON THE DISTRIBUTED SYSTEMS". In: *International Journal of Computing* 19.2 (June 14, 2020), pp. 190–198. DOI: 10.31891/1727-6209/2020/19/2-190-198.

[25] Francesco Di Cosmo. *Verification of Sometimes Termination of Lazy-Bounded Declarative Distributed Systems*. Jan. 1, 2023. DOI: 10.48550/arxiv.2308.10007.

[26] Francisco Ferreira da Silva and Stephanie Wehner. "Entanglement improves coordination in distributed systems". In: *Proceedings of the 2nd Workshop on Quantum Networks and Distributed Quantum Computing*. ACM, Sept. 8, 2025, pp. 14–20. DOI: 10.1145/3749096.3750029.

[27] "Demystifying Distributed Systems in Cloud-Native Environments". In: *Journal of Computational Analysis and Applications* 34.12 (Oct. 25, 2025). DOI: 10.48047/jocaaa.2025.34.12.43.

[28] Aziz Moumen, Mustapha Jammoukh, Laidi Zahiri, and Khalifa Mansouri. "Mechanical and Morphological Study of a Polymer Reinforced by an Animal Bio Load Using the Finite Element Method". In: Springer International Publishing, Feb. 7, 2022, pp. 281–292. DOI: 10.1007/978-3-030-90633-7_24.

[29] Mattia Zaccarini, Filippo Poltronieri, Davide Borsatti, Walter Cerroni, Luca Foschini, Genady Ya. Grabarnik, Domenico Scotece, Larisa Shwartz, Cesare Stefanelli, and Mauro

Tortonesi. “Chaos Engineering Based Kubernetes Pod Rescheduling Through Deep Sets and Reinforcement Learning”. In: *NOMS 2025-2025 IEEE Network Operations and Management Symposium*. IEEE, May 12, 2025, pp. 1–7. DOI: 10.1109/noms57970.2025.11073590.

[30] Jure Slak and Gregor Kosec. “FAST GENERATION OF VARIABLE DENSITY NODE DISTRIBUTIONS FOR MESH-FREE METHODS”. In: *WIT transactions on engineering sciences* 122 (Sept. 11, 2018), pp. 163–173.

[31] Andrea Morichetta et al. “inCoord: Intent-based Coordination in the Multi-domain Cloud-Edge Continuum”. In: *2025 IEEE 33rd International Conference on Network Protocols (ICNP)*. IEEE, Sept. 22, 2025, pp. 1–6. DOI: 10.1109/icnp65844.2025.11192462.

[32] Filippo Poltronieri, Cesare Stefanelli, and Mauro Tortonesi. “Value-of-Information Middleware Solutions for Fog and Edge Computing”. In: *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, Apr. 25, 2022, pp. 1–6. DOI: 10.1109/noms54207.2022.9789944.

[33] R Chandrasekar and T Srinivasan. “An improved probabilistic ant based clustering for distributed databases”. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI*. 2007, pp. 2701–2706.

[34] George B. Nardes, Luiz F. H. Duarte, Wesley Grignani, Douglas R. Melo, and Cesar Albenes Zeferino. “Energy-Efficient CNN Hardware Accelerator for Real-Time Construction Progress Monitoring”. In: *IEEE Access* (Jan. 1, 2025), pp. 1–1.

[35] Jiacheng Shi, Yang Yu, Jinyu Gu, and Yubin Xia. “A Hardware-Software Co-Design for Efficient Secure Containers”. In: *Proceedings of the Twentieth European Conference on Computer Systems*. ACM, Mar. 30, 2025, pp. 1229–1245. DOI: 10.1145/3689031.3717473.

[36] Mauricio Witter and Antonio Rodrigo De Vit. *Blockchain and Distributed Systems: Basic Concepts and Implications*. Mar. 26, 2024. DOI: 10.20944/preprints202403.1467.v1.

[37] Chenhui Ji, Dingji Li, Zeyu Mi, and Yubin Xia. “PrometheusMigrate: Efficient Live Migration of Confidential Virtual Machine with Software Abstraction”. In: *2024 IEEE International Conference on Joint Cloud Computing (JCC)*. IEEE, July 15, 2024, pp. 1–8. DOI: 10.1109/jcc62314.2024.00008.

[38] David Heye, Sahi Islam, Jan Pennekamp, and Klaus Wehrle. “Poster: Transport Security Orchestration Using DNS”. In: *2025 IEEE 33rd International Conference on Network Protocols (ICNP)*. IEEE, Sept. 22, 2025, pp. 1–3. DOI: 10.1109/icnp65844.2025.11192410.

[39] Héctor Benítez-Pérez, Jorge L. Ortega-Arjona, Paul E. Méndez-Monroy, Ernesto Rubio-Acosta, and Oscar A. Esquivel-Flores. “Distributed Systems Modelling”. In: Germany: Springer International Publishing, Aug. 1, 2018, pp. 41–82. DOI: 10.1007/978-3-319-97044-8_3.

[40] Ningke Li, Shenao Wang, Mingxi Feng, Kailong Wang, Meizhen Wang, and Haoyu Wang. “MalWuKong: Towards Fast, Accurate, and Multilingual Detection of Malicious Code Poisoning in OSS Supply Chains”. In: *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, Sept. 11, 2023, pp. 1993–2005. DOI: 10.1109/ase56229.2023.00073.

[41] Zhiyuan Dong, Zhaoguo Wang, Xiaodong Zhang, Xian Xu, Changgeng Zhao, Haibo Chen, Aurojit Panda, and Jinyang Li. “Fine-Grained Re-Execution for Efficient Batched Commit of Distributed Transactions”. In: *Proceedings of the VLDB Endowment* 16.8 (June 22, 2023), pp. 1930–1943. DOI: 10.14778/3594512.3594523.

[42] Anna Lackinger, Pantelis A. Frangoudis, Ivan Čilić, Alireza Furutanpey, Ilir Murturi, Ivana Podnar Žarko, and Schahram Dustdar. “Inference Load-Aware Orchestration for

Hierarchical Federated Learning". In: *2024 IEEE 49th Conference on Local Computer Networks (LCN)*. IEEE, Oct. 8, 2024, pp. 1–9. DOI: 10.1109/lcn60385.2024.10639809.

[43] Chuzhe Tang, Zhaoguo Wang, Jinyang Li, and Haibo Chen. "Sonata: Multi-Database Transactions Made Fast and Serializable". In: *Proceedings of the VLDB Endowment* 18.10 (Sept. 4, 2025), pp. 3449–3462. DOI: 10.14778/3748191.3748207.

[44] Yogesh Simmhan. "Pedagogic Practices for Teaching Distributed Systems Courses". In: *Proceedings of the 15th Annual ACM India Compute Conference*. ACM, Nov. 9, 2022, pp. 6–6. DOI: 10.1145/3561833.3568497.

[45] R Chandrasekar, RK Suresh, and SG Ponnambalam. "Evaluating an obstacle avoidance strategy to ant colony optimization algorithm for classification in event logs". In: *2006 International Conference on Advanced Computing and Communications*. IEEE. 2006, pp. 628–629.

[46] Xu Zhao, Zhiwei Lei, Guigang Zhang, Yong Zhang, and Chunxiao Xing. "WISA - Blockchain and Distributed System". In: Germany: Springer International Publishing, Sept. 22, 2020, pp. 629–641. DOI: 10.1007/978-3-030-60029-7_56.

[47] Fahim Ansari and Anis Afzal. "A Grid-Connected Distributed System for PV System". In: Springer Singapore, Jan. 25, 2021, pp. 919–924. DOI: 10.1007/978-981-15-8443-5_78.

[48] Malte Singerhoff and Torben Weis. "XTRTimeS — eXplainable Transformer for Robust Time Series Forecasting". In: *2025 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE, Mar. 17, 2025, pp. 618–619. DOI: 10.1109/percomworkshops65533.2025.00144.

[49] M. Witter and A. Rodrigo De Vit. *Blockchain e Sistemas Distribuídos: conceitos básicos e implicações*. Mar. 21, 2024. DOI: 10.48550/arxiv.2403.14854.

[50] Benedikt Full, Johannes Manner, Sebastian Böhm, and Guido Wirtz. "MicroStream vs. JPA: An Empirical Investigation". In: Germany: Springer International Publishing, Oct. 1, 2022, pp. 99–118. DOI: 10.1007/978-3-031-18304-1_6.

[51] Stephan Kleber and Frank Kargl. "Refining Network Message Segmentation with Principal Component Analysis". In: *2022 IEEE Conference on Communications and Network Security (CNS)*. IEEE, Oct. 3, 2022, pp. 281–289. DOI: 10.1109/cns56114.2022.9947242.

[52] Kamila Orynbekova, Shirali Kadyrov, Andrey Bogdanchikov, and Saidakmal Oktamov. "A novel recommender system for adapting single machine problems to distributed systems within MapReduce". In: *Bulletin of Electrical Engineering and Informatics* 14.1 (Feb. 1, 2025), pp. 687–695. DOI: 10.11591/eei.v14i1.8370.

[53] Benedikt Bollig and Paul Gastin. *Non-Sequential Theory of Distributed Systems*. Jan. 1, 2019. DOI: 10.48550/arxiv.1904.06942.

[54] Luca Di Stefano, Rocco De Nicola, and Omar Inverso. "Verification of Distributed Systems via Sequential Emulation". In: *ACM Transactions on Software Engineering and Methodology* 31.3 (Mar. 7, 2022), pp. 1–41. DOI: 10.1145/3490387.

[55] Xiangyu Gao, Yuanyuan He, Yuan Zhang, Haoyuan Zhao, Zhe Sun, Xiaoguang Li, Pengfei Liu, and Deqing Zou. "Private Set Intersection-Based Ad Click Detection". In: *2025 11th IEEE International Conference on Privacy Computing and Data Security (PCDS)*. IEEE, Aug. 1, 2025, pp. 330–336. DOI: 10.1109/pcds65695.2025.00052.

[56] "Hybrid Cloud Deployments for Distributed Systems". In: *International Journal of Computer Applications Technology and Research* (Jan. 11, 2025). DOI: 10.7753/ijcatr1401.1008.

[57] Pooja Dehankar and Susanta Das. *Navigating Trust in Distributed Systems*. Nov. 7, 2025. DOI: 10.1002/9781394288038.ch11.

[58] Rahul Malik, Chandrasekar Ramachandran, Indranil Gupta, and Klara Nahrstedt. "Samera: a scalable and memory-efficient feature extraction algorithm for short 3D video segments." In: *IMMERSCOM*. 2009, p. 18.

[59] H. El Fazazi, Mouhcine Elgarej, M. Qbadou, and Khalifa Mansouri. "Design of an Adaptive e-Learning System based on Multi-Agent Approach and Reinforcement Learning". In: *Engineering, Technology & Applied Science Research* 11.1 (Feb. 6, 2021), pp. 6637–6644. DOI: 10.48084/etasr.3905.

[60] George P Pinto, Praveen Kumar Donta, Schahram Dustdar, and Cássio Prazeres. "A Systematic Review on Privacy-Aware IoT Personal Data Stores." In: *Sensors (Basel, Switzerland)* 24.7 (Mar. 29, 2024), pp. 2197–2197. DOI: 10.3390/s24072197.

[61] Markus Dahlmanns and Klaus Wehrle. "Protocol Security in the Industrial Internet of Things". In: *NOMS 2024-2024 IEEE Network Operations and Management Symposium*. IEEE, May 6, 2024, pp. 1–4. DOI: 10.1109/noms59830.2024.10575096.