Monteinstitute : Pages:1–15

Original Research

Decentralized Schema Evolution Management via Collective Learning in Enterprise Data Ingestion Pipelines

Syed Faizan Ali[1] and Muhammad Haroon Javed[2]

[1]Department of Computer Science, University of Baltistan, Campus Road, Skardu 19110, Gilgit-Baltistan, Pakistan.
[2] Department of Business Administration, Southern Business School, University of Southern Punjab, Bosan Road, Multan 60000, Pakistan.

Abstract

Enterprise data ingestion pipelines are increasingly required to integrate heterogeneous data sources whose schemas evolve continuously and often independently. Conventional schema management practices rely on centralized registries and tightly controlled governance processes, which become difficult to sustain when organizations adopt decentralized ownership models and rapidly changing application ecosystems. At the same time, large organizations seek to maintain global consistency constraints, auditability, and reliability of downstream analytics while allowing local teams to introduce changes at their own pace. This tension between local autonomy and global coherence motivates approaches that distribute responsibility for schema evolution while still coordinating decisions across multiple ingestion components. This paper examines a decentralized perspective on schema evolution management for enterprise ingestion pipelines, with particular emphasis on collective learning among ingestion agents and services. The proposed perspective models each ingestion component as a learning agent that maintains a local representation of schemas and their transformations, and that exchanges summaries of its beliefs with neighboring agents in the ingestion topology. Through this view, schema evolution becomes a continuous adaptation problem where agents collaboratively update shared representations of compatible schemas and transformation operators. The paper develops a linear modeling framework that formalizes this process and discusses how it can be instantiated in practical settings such as log aggregation, event streaming, and batch ingestion architectures. The discussion remains focused on structural properties of the models and algorithms rather than specific technologies, with the goal of supporting a range of system realizations.

1. Introduction

Enterprise data landscapes are characterized by a growing number of applications, microservices, and external platforms that continuously emit structured or semi-structured data [1]. Data ingestion pipelines are responsible for capturing these flows, converting them to internal formats, and routing them toward analytical and operational consumers. Over time, the schemas associated with such flows evolve due to new business requirements, deprecations of legacy attributes, and integration of new data sources. This phenomenon, often referred to as schema drift, poses a persistent challenge because ingestion components must adapt without breaking downstream processes or violating quality constraints [2]. In many organizations, this challenge has been addressed by establishing centralized schema registries, committees, or governance tools that mediate all schema changes and enforce global rules.

As organizations scale, centralized approaches encounter a combination of social and technical bottlenecks. From a social perspective, a central authority can become a gatekeeper that delays the onboarding of new sources, while domain teams desire autonomy to evolve their schemas in alignment with local needs. From a technical perspective, the variety of ingestion technologies, including message queues, change data capture tools, and cloud-native streaming platforms, complicates uniform enforcement of schema policies [3]. Different teams may adopt different serialization formats, naming conventions,

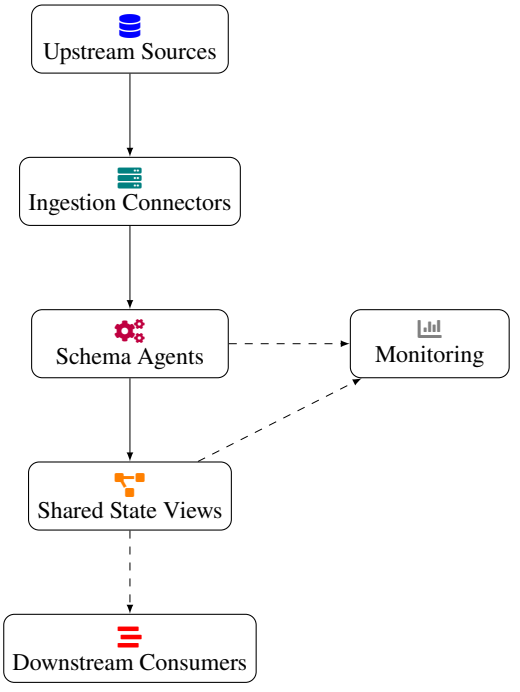

Figure 1: High-level structure of sources, ingestion connectors, schema agents, shared state views, and downstream consumers with a side monitoring component.

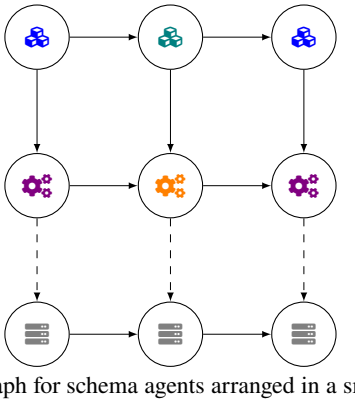

Figure 2: Example communication graph for schema agents arranged in a small lattice, illustrating local message exchange among neighboring ingestion components.

and documentation practices, all of which influence how schema evolution is perceived and controlled. These pressures lead to situations where the actual schemas encountered in ingestion jobs diverge from centrally curated specifications, and where local workarounds proliferate.

A decentralized perspective treats each ingestion component as a semi-autonomous agent with authority over a subset of data sources and transformations [4]. Rather than relying on a single global registry, the organization may rely on a distributed set of schema views that are reconciled through coordination protocols. Such a perspective is compatible with architectural trends that favor domain-oriented data ownership and self-serve data platforms, because it aligns schema management with the components that perform ingestion work. However, decentralization also raises the risk of fragmentation, duplication of effort, and accumulation of inconsistencies that can undermine trust in shared data assets.

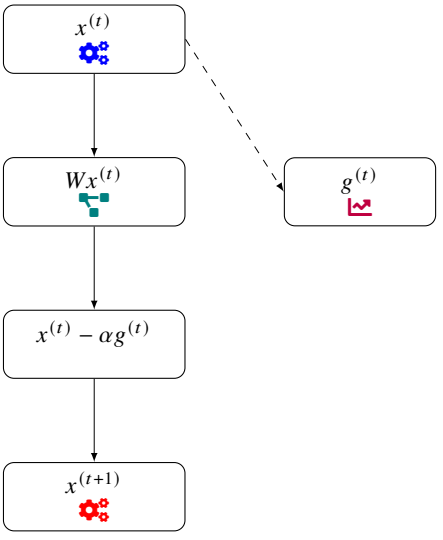

Figure 3: Conceptual view of a single collective update step combining consensus through a linear operator with a local gradient-based adjustment for schema-related state.

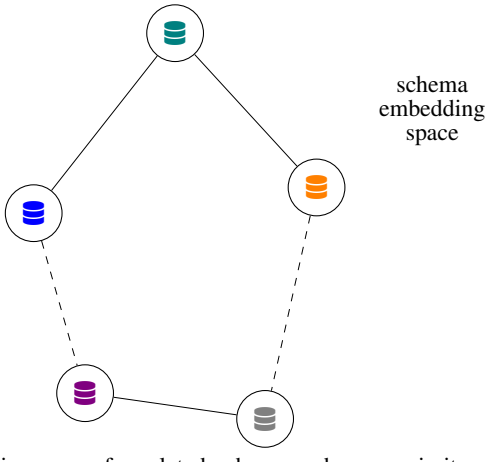

Figure 4: Illustrative embedding space for related schemas where proximity and links encode similarity and regularization structure across ingestion components.

To address these risks, decentralized approaches must incorporate mechanisms for agents to learn from one another and converge toward consistent interpretations of schemas and transformations [5] [6].

Collective learning provides a set of conceptual tools for coordinating decentralized agents without imposing strict central control. By allowing ingestion agents to exchange structured summaries of observed schemas, transformation patterns, and validation outcomes, one can view schema evolution as a distributed learning process over a communication graph. Each agent maintains internal representations that reflect local observations, but these representations are influenced over time by information from neighboring agents [7]. Linear models and operators provide a natural language to describe such processes because they can capture aggregation, consensus, and regularization effects in a mathematically transparent way. This paper develops such a linear modeling view for decentralized schema evolution management, focusing on conditions under which agents can converge toward consistent decisions about schema compatibility and transformation selection while preserving local autonomy.

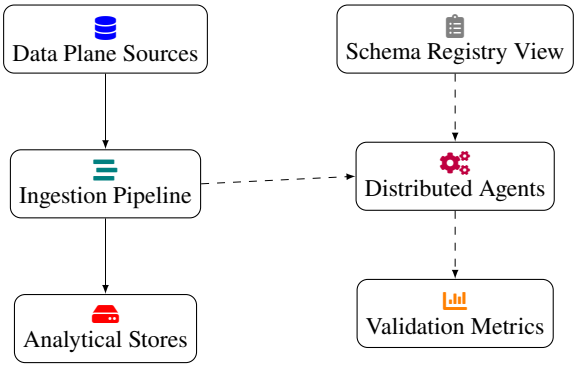

Figure 5: Separation of data plane and control mechanisms with schema-aware agents coordinating registry views, validation metrics, and ingestion behavior.

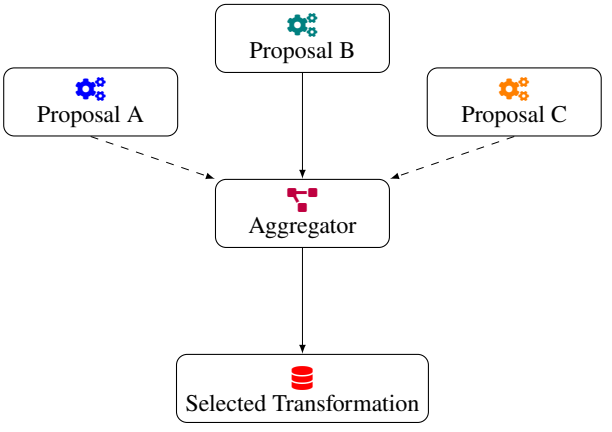

Figure 6: Aggregation of multiple candidate schema transformations into a single selected mapping through a centralized yet lightweight combining step.

2. Background and Problem Formulation

Component	Icon	Primary Role	Schema Scope
Upstream source		Emit operational records	Local application model
Ingestion connector		Capture and serialize flows	Source-specific schema
Schema agent		Track and adapt schemas	Local plus neighbor view
Shared state service		Aggregate beliefs	Multi-component view
Downstream consumer		Query or process records	Logical analytical view

Table 1: Key ingestion components and their schema-related roles.

Enterprise data ingestion pipelines typically consist of connectors that extract data from operational systems, transformation jobs that reshape or clean the data, and sinks that store the resulting records in analytical warehouses, data lakes, or operational stores. Each connector is associated with one or more logical schemas describing the structure of incoming records, such as fields for identifiers, timestamps, and domain-specific attributes [8]. In practice, these schemas are often defined by application developers under constraints that are only loosely coordinated with downstream analytical requirements. As

Aspect	Centralized management	Decentralized collective learning
Authority	Single registry team	Per-component agents
Change approval	Pre-deployment gate	Continuous adaptation
Failure impact	Single point of disruption	Localized disturbance
Knowledge locality	Global but coarse	Local but detailed
Scalability	Limited by central workflows	Aligned with topology
Latency to adapt	Higher, batch-oriented	Lower, incremental
Auditability	Registry-driven logs	Distributed agent logs

Table 2: Qualitative comparison of centralized and decentralized schema evolution regimes.

Symbol	Type	Interpretation	Origin
x_i	Vector	Local schema state of component i	Agent
W	Matrix	Consensus operator on global state	Topology
$g^{(t)}$	Vector	Stacked local gradients at iteration t	Loss signals
α	Scalar	Step size for local updates	Tuning
L	Matrix	Graph Laplacian for regularization	Connectivity
λ	Scalar	Weight of smoothness penalty	Tuning
M_i	Matrix	Local schema transformation candidate	Agent

Table 3: Core symbols used in the linear modeling formulation of schema evolution.

Topology	Degree pattern	Consensus speed	Operational example
Ring	Uniform small degree	Slow to moderate	Sequential staging jobs
Grid	Moderate degree	Moderate	Regional clusters
Star	Hub with high degree	Fast via hub	Central log router
Hierarchical	Layered degrees	Two-scale mixing	Domain hub per unit
Random sparse	Heterogeneous degrees	Data-dependent	Ad hoc service mesh

Table 4: Representative communication topologies for schema agents and qualitative convergence properties.

Parameter	Meaning	Typical range	Sensitivity
α	Local update step size	10^{-4}–10^{-1}	High for stability
λ	Graph smoothness weight	10^{-3}–10^{1}	Medium for coherence
τ	Error threshold for alerts	10^{-2}–10^{-1}	High for noise vs drift
ρ	Forgetting factor in history	0.8–0.99	Medium for reactivity
ω_i	Weight for proposal M_i	0–1	Medium for bias

Table 5: Selected tuning parameters and qualitative sensitivity in the collective learning process.

applications change, connectors observe new fields, modified types, or altered semantics that are not always communicated in a timely way to data platform teams. This misalignment gives rise to schema evolution incidents, where ingestion jobs fail, silently drop fields, or apply incorrect transformations [9].

For modeling purposes, consider a set of ingestion components indexed by an index set that can be denoted by a finite set of integers. Each component is responsible for a subset of sources and maintains its own local schema catalog. At any point in time, a component has a working schema for each of its sources, which may differ from a hypothetical global canonical schema. The system must decide how to reconcile these local schemas in order to provide consistent views to consumers [10]. One can view this

Metric	Definition sketch	Focus	Aggregation level
Convergence gap	Mean $\|x_i - \bar{x}\|_2$	Agreement	Global
Drift detection	Rate of threshold crossings	Change onset	Per source
Failure rate	Ingestion errors per volume	Robustness	Pipeline
Adaptation delay	Time from change to recovery	Responsiveness	Component
Alert precision	Valid alerts over all alerts	Signal quality	System

Table 6: Example metrics for evaluating decentralized schema evolution behavior.

Mode	Cause	Immediate effect	Longer-term risk
Silent field drop	Unrecognized addition	Data loss	Biased analytics
Type mismatch	Incompatible change	Validation fail	Pipeline rollback
Stale transformation	Outdated mapping matrix	Misrouted data	Conflicting reports
Partitioned agents	Network segmentation	Divergent states	Hard recovery
Over-aggregation	Excessive consensus weight	Loss of locality	Overfitting global pattern

Table 7: Illustrative failure modes related to schema evolution and their potential impact.

Scenario	Schema change pattern	Traffic profile	Example domain
Slow drift	Occasional field additions	Moderate, steady	Enterprise resource planning
Burst change	Short bursts of new variants	Spiky	Marketing campaigns
High churn	Frequent minor modifications	High, continuous	User telemetry
Rare but large	Infrequent structural shifts	Mixed	Regulatory reporting
Multi-tenant mix	Independent tenant updates	Heterogeneous	Shared SaaS platform

Table 8: Workload patterns influencing the design and tuning of schema evolution strategies.

Role	Example decision	Local signals	Shared signals
Connector agent	Accept new field	Parsing errors	Neighbor states
Aggregator agent	Merge schema variants	Upstream conflicts	Global summaries
Registry view	Publish canonical revision	Proposed mappings	System health
Monitoring node	Raise evolution incident	Error rates	Drift statistics
Reviewer	Approve semantic changes	Business context	Historical traces

Table 9: Typical roles in the ecosystem and the signals involved in schema evolution decisions.

as a problem of coordinating local schema evolution decisions under partial information, where each component has a limited view of the overall data landscape and yet its decisions can affect downstream users through shared sinks or intermediate topics.

A useful abstraction is to represent the ingestion pipeline as a graph whose vertices correspond to components and whose edges represent data or control dependencies. If two components share a sink, or if one component feeds another through a transformation, they are connected by an edge. This graph captures the pathways along which schema changes can propagate [11]. When a source schema changes at one component, the effects may be felt at other components connected through intermediate transformations. The problem is to design mechanisms through which components can align their schema interpretations and transformations so that propagated changes remain compatible with expectations elsewhere in the graph.

Traditional solutions rely on central registries where schemas must be registered and approved before deployment [12]. In the modeling language described here, such an arrangement corresponds to a topology where all components are connected to a special hub node that maintains the authoritative schema definitions. While this simplifies global reasoning, it creates a single point of coordination and potential failure. A decentralized approach instead assumes that no single node has complete authority. Each component maintains its own schema interpretations and updates them based on local observations and messages from neighboring components [13]. The key question is how to structure these messages and update rules so that the system exhibits a form of emergent global coherence.

To formalize this situation, one may associate with each component a state vector that encodes its beliefs about schemas and transformations. The dimension of this vector can be chosen to reflect the number of fields, types, and transformation parameters that the component must manage [14]. The global system state then becomes the concatenation of all local state vectors. Schema evolution corresponds to trajectories of this global state as components observe new data and exchange information. The problem formulation adopted in this paper is to model these trajectories using linear operators that capture both the effect of local observations and the effect of communication between components, with the aim of understanding convergence properties and trade-offs between local autonomy and global consistency.

3. Decentralized Collective Learning Framework

In a decentralized collective learning framework, each ingestion component maintains a local state that represents its current schema beliefs and transformation parameters [15]. Let the state of component indexed by an integer be denoted by a vector that lies in a Euclidean space. This state can encode, for example, embeddings of field names, types, and example values, as well as parameters for transformation functions that map source schemas to target schemas. The global system state is the stacked vector that collects all local states [16]. The evolution of this global state is driven by two kinds of influences: local updates based on newly observed records and communication updates based on messages received from neighboring components in the ingestion graph.

A simple yet expressive way to capture communication is through a linear consensus operator. One may define a matrix whose entries encode how strongly each component influences another during a communication step. The matrix can be taken as row-stochastic, so that each row corresponds to a convex combination of neighbor states [17]. A communication round can then be written as

$$x^{(t+1)} = W x^{(t)},$$

where the vector collects all local states at iteration index. Under standard assumptions on the connectivity of the underlying graph and the weights, repeated application of this update can drive the states toward a consensus subspace [18]. However, consensus alone does not address the need to incorporate local schema observations, which can cause components to adjust their states in ways that may temporarily disagree with neighbors.

To incorporate local learning, one can augment the consensus dynamics with a term that represents the effect of gradient-like updates based on local error signals. Suppose each component tries to minimize a local loss that measures the discrepancy between its predicted representation of a schema and the representation implied by observed data or by local configuration changes. A generic gradient step for the global state can be written as [19]

$$x^{(t+1)} = x^{(t)} - \alpha g^{(t)},$$

where the vector concatenates local gradients and the scalar controls the step size. Combining consensus and local gradients leads to an update of the form

$$x^{(t+1)} = W x^{(t)} - \alpha g^{(t)}.$$

This linear-affine form captures the idea that components move their states partly toward neighbor averages and partly in the direction suggested by local observations. Under suitable conditions on the matrix and step size, such dynamics can converge to a fixed point that balances local objectives with agreement among neighbors [20].

Schema evolution incidents correspond to situations where local gradients become large because observed records deviate from expectations. For instance, the appearance of a new field or a type change can cause a local loss to spike, prompting the component to adjust its state. Through the communication term, this adjustment partially propagates to neighbors, which may respond by updating their own states [21]. If the system is well tuned, these cascades of adjustments should dampen over time as components settle on new representations of schemas and transformations that are compatible with one another. If communication is too weak or too infrequent, local adjustments may fail to propagate and inconsistencies may persist. If communication is too strong, local anomalies may spread widely before being validated, leading to instability.

The collective learning framework also provides a way to reason about different roles that components may play in the ingestion topology [22]. Some components may act as aggregators that receive data from many sources and redistribute it to multiple sinks. Others may be leaf nodes that connect to a single source or sink. The communication matrix can reflect these structural differences by assigning different weights or degrees of connectivity [23]. For example, an aggregator may place more weight on messages from leaf nodes when updating its view of upstream schemas, while leaf nodes may place more weight on the aggregator when updating their understanding of downstream expectations. These asymmetries can be encoded in the rows of the matrix and analyzed using spectral properties of the associated linear operators.

Finally, the framework suggests designing communication messages as low-dimensional summaries of schema beliefs rather than full schema definitions. Rather than exchanging complete schema documents, components can exchange vectors or matrices that encode embeddings of schema elements, estimates of transformation stability, or statistics about validation errors [24]. Such summaries can be updated using linear transformations that map local representations into message space and back. For example, if a component maintains a state vector, it can compute a message vector using a matrix multiplication of the form

$$m_i^{(t)} = H_i x_i^{(t)},$$

where the matrix encodes how local state is projected into a shared message space [25]. Neighbors receiving this message can integrate it into their own updates using similar linear operators. This design makes it possible to implement collective learning protocols that are efficient in network bandwidth while remaining amenable to analysis using linear algebra.

4. Linear Models for Schema Evolution and Conflict Resolution

Representing schemas and schema transformations in a linear space enables the use of linear models to track and resolve schema evolution. Consider a space of dimension where vectors encode schema-related features, such as field presence indicators, type encodings, and learned embeddings for semantic similarity [26]. Each component maintains a schema vector for each source it ingests. For a given source and component, denote this vector by a symbol that lies in the vector space. A canonical or target schema for the same logical entity can be represented by another vector [27]. The discrepancy between the local and canonical views can then be quantified by a norm such as the Euclidean norm, written as

$$e_i = \|s_i - c_i\|_2.$$

When the discrepancy grows beyond a threshold, the component may infer that a schema evolution event has occurred and initiate an adaptation process.

Adaptation can be modeled as a linear transformation that maps old schemas to new ones [28]. Suppose that a component maintains a transformation matrix for a given source that maps raw schema vectors to a standardized representation used downstream. When a schema evolves, the component must adjust this matrix so that transformed data remains compatible with downstream expectations. One can formulate this as a least-squares problem where the goal is to find a matrix that minimizes discrepancies between transformed feature vectors and desired targets. If a set of feature-target pairs is available, with a matrix collecting feature vectors and a matrix collecting targets, one can consider the objective [29]

$$J(M) = \|MF - T\|_F^2,$$

where the Frobenius norm measures squared reconstruction error. Under this formulation, an updated transformation matrix can be obtained by solving the associated normal equations or by applying iterative methods that operate using matrix-vector products [30].

Conflict resolution arises when multiple components propose different interpretations or transformations for the same logical schema. In a decentralized setting, no single authority can dictate the correct version, so conflicts must be resolved through some form of aggregation. Linear models provide one approach by defining a global transformation as a weighted combination of local candidates. If component proposals for a transformation are matrices, and associated weights sum to one, a simple aggregator can be written as [31]

$$M^\star = \sum_i \omega_i M_i,$$

where the sum runs over components that contribute proposals and the weights encode trust or relevance. The aggregated matrix defines a consensus transformation that can be communicated back to components. This approach can be refined by learning the weights based on historical performance or domain-specific rules, while retaining a linear form that facilitates analysis [32].

Graph-based regularization offers another linear mechanism for aligning schema representations across components. Treat the ingestion topology as a graph with an associated Laplacian matrix. For a stack of schema vectors collected into a matrix, a smoothness penalty of the form

$$R(S) = \operatorname{tr}(S^\top L S)$$

encourages neighboring components to maintain similar schema representations [33]. Minimizing an objective that combines local reconstruction error with such a regularizer yields an optimization problem of the form

$$\min_S \ \|S - \hat{S}\|_F^2 + \lambda \operatorname{tr}(S^\top L S),$$

where the matrix collects locally preferred schema vectors and the scalar controls the strength of regularization. The optimal solution satisfies a linear system whose structure reflects the graph connectivity [34]. This provides a principled way to propagate schema adjustments across components while penalizing sharp differences along graph edges.

Linear models can also be used to approximate compatibility constraints between schemas. Suppose that compatibility between two schemas is captured by a scalar score that is close to one when they are compatible and close to zero otherwise. A linear scoring model can be defined by embedding each schema into a vector and setting the score to be the inner product of transformed embeddings [35]. If a matrix maps embeddings into a scoring space, compatibility can be computed as

$$\gamma_{ij} = u^\top V(s_i - s_j),$$

where the vector and matrix define the scoring direction and transformation. Training such a model involves adjusting parameters so that observed compatible pairs yield high scores while incompatible pairs yield low scores [36]. In a decentralized setting, components can maintain local estimates of these

parameters and update them using shared compatibility judgments, leading to a distributed linear model that reflects the collective experience of the ingestion system.

5. Implementation and Evaluation Considerations

Implementing decentralized schema evolution management based on collective learning requires integrating the linear models and update rules into actual ingestion infrastructure. Each ingestion component must be augmented with a schema agent capable of maintaining state vectors, computing updates based on data observations, and exchanging messages with neighbors. In practice, these agents can be embedded into existing connector frameworks or transformation services, exposing interfaces for observing schema-related events such as new fields, validation failures, or transformation errors [37]. The agents translate such events into updates of their internal state vectors using predefined feature encodings and linear update rules. Care must be taken to ensure that such computations remain lightweight so that ingestion throughput and latency requirements are not compromised.

Communication between agents can be implemented over the same messaging fabric that carries data or over a separate control plane. Messages contain compact representations of schema states, such as low-dimensional embeddings or transformation summaries, as predicted by linear projections [38]. To limit bandwidth usage, one can restrict communication to events where local discrepancy measures exceed thresholds or when components detect significant changes in upstream or downstream behavior. The frequency of communication steps relative to data ingestion steps influences the degree of coordination achievable. If communication is infrequent, agents may rely more heavily on local observations, leading to slower convergence to consistent interpretations [39]. If communication is frequent, the system may track changes more quickly but at the cost of additional overhead.

Evaluation of a decentralized schema evolution framework involves both quantitative and qualitative dimensions. Quantitatively, one may measure the rate at which agents converge to consistent schema representations after evolution events, the frequency of ingestion failures or validation errors, and the volume of corrective actions required by human operators. Synthetic workloads can be constructed by generating sequences of schema changes and simulating data streams under different topologies and communication protocols [40]. In such simulations, the linear dynamics of the collective learning process can be directly observed by tracking norms of differences between local state vectors. Metrics such as average pairwise distance between schema representations or variance of transformation parameters across components provide indicators of global coherence.

Qualitative evaluation focuses on how the framework interacts with organizational practices around data ownership and governance [41]. Even though the models are linear and mathematically tractable, their behavior must be interpretable by engineers and data stewards who are responsible for operational decisions. Providing visualizations of the learned schema embeddings and their evolution over time can help stakeholders understand how the system responds to changes. Exposing simple controls, such as parameters that adjust the strength of regularization or the weight given to neighbor messages, allows teams to tune the balance between local autonomy and global coordination according to their needs. These controls can be mapped directly to scalar parameters in the linear models, making it easier to predict how adjusting them will affect system behavior [42].

Practical deployments must also address failure modes and operational constraints. Network partitions or component outages can interrupt communication among agents, leading to temporary divergences in schema beliefs. The linear framework can accommodate such situations by treating missing messages as implicit identity updates, where components retain their current state [43]. When connectivity is restored, the consensus dynamics naturally work to reduce accumulated discrepancies. However, care is required to avoid sudden global shifts in schema representations that could surprise downstream consumers. Techniques such as damping, where communication weights are temporarily reduced after reconnection, can be modeled by adjusting the entries of the communication matrix over time.

Finally, any evaluation must account for interaction with existing governance artifacts such as documentation, data catalogs, and access control policies [44]. The linear models and collective learning

mechanisms do not replace these artifacts but rather provide an additional layer of coordination in the operational plane. For example, an organization might still maintain a human-curated catalog of canonical schemas, while allowing decentralized agents to propose updates and transformations that are later reviewed. In such a configuration, the canonical catalog can be viewed as defining reference vectors in the schema embedding space, and the linear models can compute deviations from these reference points [45]. Observed discrepancies can trigger alerts or review workflows, creating a feedback loop between automated collective learning and human oversight.

6. Discussion and Limitations

Adopting a decentralized, collectively learned approach to schema evolution management introduces both opportunities and limitations. On the one hand, modeling ingestion components as learning agents with linear update rules offers a structured way to balance local autonomy with global coherence. On the other hand, the abstractions required to make the models tractable inevitably simplify the complex semantics of real-world schemas [46]. For instance, representing schemas as fixed-dimensional vectors assumes that all relevant aspects of structure and meaning can be captured in this form. In practice, schemas may include nested structures, conditional fields, or context-dependent semantics that are difficult to encode linearly without loss of nuance.

Another limitation concerns the reliance on linear dynamics for communication and adaptation. Linear consensus and regularization operators provide valuable insight into convergence behavior, but real ingestion systems may exhibit nonlinearities arising from threshold-based decisions, rule-based validations, or conditional transformations [47]. In some cases, linear models can approximate these behaviors around operating points or in aggregate, but discrepancies may arise under large disturbances or rare events. Extending the framework to incorporate piecewise linear or nonlinear components could improve fidelity but would complicate analysis. The choice between analytical tractability and representational richness must be made with awareness of the specific requirements and risk tolerance of the organization [48].

The effectiveness of collective learning also depends on the quality and quantity of observations available to each agent. In environments where data volumes are low or changes occur infrequently, agents may have limited evidence from which to infer schema evolution, leading to slow or uncertain updates. Conversely, in high-volume environments with frequent changes, agents may be overwhelmed by signals, making it difficult to distinguish transient anomalies from persistent new patterns. Linear models can incorporate temporal smoothing or forgetting factors, but setting these parameters appropriately is a nontrivial task [49]. Different domains, such as financial transactions versus user interaction logs, may require different trade-offs between responsiveness and stability.

Furthermore, decentralization introduces governance questions about responsibility and accountability. When schema evolution decisions emerge from the interaction of multiple agents rather than from a centralized committee, it can be harder to attribute specific outcomes to particular actors [50]. While logs of agent states and messages can provide an audit trail, interpreting these records requires an understanding of the underlying models and their parameters. Organizations may need to invest in tooling and processes that make the behavior of collective learning mechanisms transparent to stakeholders, so that issues can be diagnosed and addressed in a timely manner. This transparency requirement may influence the choice of model complexity and the way in which parameters are exposed for inspection.

Finally, there are limits to what any automated schema evolution framework can achieve without human involvement [51]. Certain schema changes reflect deep shifts in business meaning that cannot be reliably inferred from statistical patterns alone. For example, a field might be repurposed to represent a different concept, or a new combination of fields might implement a revised regulatory definition. In such cases, human decision makers must specify the appropriate semantics and transformations [52]. A decentralized collective learning system can support these decisions by providing aggregated views of current usage patterns and by propagating approved changes efficiently. However, it cannot replace

domain expertise. Recognizing these boundaries is important for setting realistic expectations and for integrating the framework into broader governance practices.

7. Conclusion

This paper has examined a decentralized approach to schema evolution management in enterprise data ingestion pipelines, framed in terms of collective learning among ingestion components [53]. By representing schemas, transformations, and compatibility assessments as vectors and matrices, the discussion has outlined how linear models can capture both local adaptation to schema changes and global coordination through communication among agents. The resulting framework portrays schema evolution as a trajectory of a global system state governed by linear-affine dynamics that combine consensus with gradient-like updates derived from local observations.

Within this perspective, each ingestion component becomes a learning agent that contributes to and benefits from shared knowledge about schemas and transformations [54]. Communication protocols defined by matrices and projection operators enable agents to exchange compact summaries of their beliefs, while regularization terms based on graph structure encourage alignment of schema representations across the ingestion topology. These mechanisms can be implemented with modest overhead by embedding schema agents within existing ingestion infrastructure and by leveraging available messaging channels for control communication. Evaluation considerations highlight the need to monitor convergence, coherence, and failure modes, as well as to integrate automated mechanisms with human governance processes.

At the same time, several limitations and open questions remain [55]. Linear models offer analytical clarity but may not capture all the nuances of real-world schema semantics and operational constraints. The choice of feature encodings, communication topologies, and parameter settings can significantly influence outcomes and may need to be tuned for specific organizational contexts. Human oversight remains essential for interpreting and guiding schema evolution decisions that carry significant business or regulatory implications. Future explorations may consider hybrid approaches that combine linear collective learning with richer semantic models, as well as empirical studies that assess the behavior of such frameworks in large-scale production environments [56].

References

[1] H. Duan, S. Shao, B. Su, and L. Zhang, "New development thoughts on the bio-inspired intelligence based control for unmanned combat aerial vehicle," *Science China Technological Sciences*, vol. 53, pp. 2025–2031, 7 2010.

[2] D. H. Kim, "Self-organization of unicycle swarm robots based on a modified particle swarm framework," *International Journal of Control, Automation and Systems*, vol. 8, pp. 622–629, 6 2010.

[3] Q. kun Song, M. meng Xu, and Y. Liu, "Rbf neural network controller research based on afsa algorithm," *International Journal of Hybrid Information Technology*, vol. 7, pp. 33–38, 5 2014.

[4] V. Vijaykumar, R. Chandrasekar, and T. Srinivasan, "An ant odor analysis approach to the ant colony optimization algorithm for data-aggregation in wireless sensor networks," in *2006 International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–4, IEEE, 2006.

[5] M.-P. Li and M. Yao, "An optimized ant system for clustering with elitist ant and local search," *ITM Web of Conferences*, vol. 7, pp. 05011–, 11 2016.

[6] H. SH, "Building a dynamic data ingestion framework to manage schema evolution for an enterprise," *INTERNATIONAL JOURNAL*, vol. 4, no. 2, 2022.

[7] O. Rohat and D. Popescu, "Web system for the remote control and execution of an iec 61499 application," *Studies in Informatics and Control*, vol. 23, 9 2014.

[8] null Pissa Sandhya Rani and null K. S. V. Phani Kumar, "A review on optimal generation techniques used in microgrids," *International Journal of Engineering Research and*, vol. V4, 10 2015.

[9] M. Pavone, R. A. Ramadan, and A. V. Vasilakos, "Intelligent cloud computing," *Memetic Computing*, vol. 8, pp. 253–254, 11 2016.

[10] C. V. Raghavendran, G. N. Satish, and P. S. Varma, "Intelligent routing techniques for mobile ad hoc networks using swarm intelligence," *International Journal of Intelligent Systems and Applications*, vol. 5, pp. 81–89, 12 2012.

[11] X. Lei, C. Ying, F.-X. Wu, and J. Xu, "Clustering ppi data by combining fa and shc method," *BMC genomics*, vol. 16, pp. 1–10, 1 2015.

[12] D. Karaboga and B. Akay, "A survey: algorithms simulating bee swarm intelligence," *Artificial Intelligence Review*, vol. 31, pp. 61–85, 10 2009.

[13] 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*, pp. 2701–2706, 2007.

[14] X. Wang and W. Tang, "Fuzzy epq inventory models with backorder," *Journal of Systems Science and Complexity*, vol. 22, pp. 313–323, 5 2009.

[15] K. AshokBabu, D. S. Rao, and S. Lakshminarayana, "Swarm intelligence based energy efficient routing protocol for wireless ad-hoc networks," *International Journal of Computer Applications*, vol. 62, pp. 34–39, 1 2013.

[16] I.-D. Psychas, M. Schauer, J.-U. Böhrnsen, M. Marinaki, Y. Marinakis, S. C. Langer, and G. E. Stavroulakis, "Detection of defective pile geometries using a coupled fem/sbfem approach and an ant colony classification algorithm," *Acta Mechanica*, vol. 227, pp. 1279–1291, 1 2016.

[17] Y. Wang, Q. Wang, H. Zhang, A. Kang, Y. Xia, and Y. Sun, "Improved particle swarm optimization algorithm for optimization of power communication network," *International Journal of Grid and Distributed Computing*, vol. 9, pp. 225–236, 1 2016.

[18] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Inspiration for optimization from social insect behaviour," *Nature*, vol. 406, pp. 39–42, 7 2000.

[19] M. I. Razzak, M. K. Khan, K. Alghathbar, and J. H. Park, "Energy efficient distributed face recognition in wireless sensor network," *Wireless Personal Communications*, vol. 60, pp. 571–582, 4 2011.

[20] M. Ashwin, S. Kamalraj, and M. Azath, "Multi objective trust optimization for efficient communication in wireless m learning applications," *Cluster Computing*, vol. 22, pp. 10687–10695, 9 2017.

[21] R. Chandrasekar, V. Vijaykumar, and T. Srinivasan, "Probabilistic ant based clustering for distributed databases," in *2006 3rd International IEEE Conference Intelligent Systems*, pp. 538–545, IEEE, 2006.

[22] M. Manafifard, H. Ebadi, and H. A. Moghaddam, "Multi-player detection in soccer broadcast videos using a blob-guided particle swarm optimization method," *Multimedia Tools and Applications*, vol. 76, pp. 12251–12280, 5 2016.

[23] S. Chen, Y.-J. Zheng, C. Cattani, and W. Wang, "Modeling of biological intelligence for scm system optimization," *Computational and mathematical methods in medicine*, vol. 2012, pp. 769702–769702, 11 2011.

[24] K. H. Leong, H. Abdul-Rahman, C. Wang, C. C. Onn, and S. C. Loo, "Bee inspired novel optimization algorithm and mathematical model for effective and efficient route planning in railway system.," *PloS one*, vol. 11, pp. e0166064–, 12 2016.

[25] K. Ramalakshmi, P. K. Sasikumar, and P. G. Scholar, "Study on security and quality of service implementations in p2p overlay network for efficient content distribution," *International Journal of Research in Engineering and Technology*, vol. 03, pp. 455–464, 4 2014.

[26] R. T. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intelligence*, vol. 2, pp. 189–208, 8 2008.

[27] S. Hauert, L. Winkler, J.-C. Zufferey, and D. Floreano, "Ant-based swarming with positionless micro air vehicles for communication relay," *Swarm Intelligence*, vol. 2, pp. 167–188, 8 2008.

[28] S. K. E, S. M. Kusuma, and B. P. V. Kumar, "Clustering protocol for wireless sensor networks based on rhesus macaque (macaca mulatta) animal's social behavior," *International Journal of Computer Applications*, vol. 87, pp. 20–27, 2 2014.

[29] V. 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*, pp. 1–6, 2006.

[30] M. Ismail and I. El-Henawy, "A comparison between fppso and b&b algorithm for solving integer programming problems," *International Journal of Computer Applications Technology and Research*, vol. 4, pp. 197–201, 3 2015.

[31] H. Qian, Y. Chen, Y. Sun, N. Liu, N. Ding, Y. Xu, G. Xu, Y. Tang, and J. Yan, "Vehicle safety enhancement system: Sensing and communication:," *International Journal of Distributed Sensor Networks*, vol. 9, pp. 542891–, 12 2013.

[32] P.-W. Tsai, A. Alsaedi, T. Hayat, and C.-W. Chen, "A novel control algorithm for interaction between surface waves and a permeable floating structure," *China Ocean Engineering*, vol. 30, pp. 161–176, 4 2016.

[33] B. Niu, H. Wang, Q. Duan, and L. Li, "Biomimicry of quorum sensing using bacterial lifecycle model," *BMC bioinformatics*, vol. 14, pp. 1–13, 5 2013.

[34] V. Kumar, S. Agarwal, and A. Singh, "Intrusion detection techniques based on cross layer for wireless local area networks," *INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY*, vol. 5, pp. 94–97, 7 2013.

[35] H. Wang, L. Liao, D. Wang, S. Wen, and M. Deng, "Improved artificial bee colony algorithm and its application in lqr controller optimization," *Mathematical Problems in Engineering*, vol. 2014, pp. 1–8, 4 2014.

[36] V. Agrawal, R. Rastogi, and D. C. Tiwari, "Spider monkey optimization: a survey," *International Journal of System Assurance Engineering and Management*, vol. 9, pp. 929–941, 11 2017.

[37] C. Ramachandran, R. Malik, X. Jin, J. Gao, K. Nahrstedt, and J. Han, "Videomule: a consensus learning approach to multi-label classification from noisy user-generated videos," in *Proceedings of the 17th ACM international conference on Multimedia*, pp. 721–724, 2009.

[38] F. Wan, W. Yuan, and J. Zhou, "Derivation of tri-level programming model for multi-reservoir optimal operation in inter-basin transfer-diversion-supply project," *Water Resources Management*, vol. 31, pp. 479–494, 12 2016.

[39] P. Maji, S. K. Pal, and A. Skowron, "Special issue of natural computing on pattern recognition and mining - preface: pattern recognition and mining," *Natural Computing*, vol. 15, pp. 355–357, 2 2015.

[40] M. Effatparvar, S. Hoseinpour, and V. Asadzadeh, "Resource allocation in computational grids environment using improved particle swarm optimization algorithm," *International Journal of Computer Applications Technology and Research*, vol. 3, pp. 529–532, 8 2014.

[41] Y. Zhou, J. Duan, and L. Shao, "Application of artificial fish swarm algorithm in radial basis function neural network," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 14, pp. 699–706, 6 2016.

[42] mudassir makhdoomi, "Data mining approach for big data analysis:a theoritical discourse," *International Journal of Advanced Research in Computer Science*, vol. 8, pp. 104–109, 8 2017.

[43] I. Susnea, G. Vasiliu, and D. E. Mitu, "Enabling self-organization of the educational content in ad hoc learning networks," *Studies in Informatics and Control*, vol. 22, 6 2013.

[44] W. Ke, Z. ping Peng, Q. Yuan, B. rong Hong, K. Chen, and Z. Cai, "A method of task allocation and automated negotiation for multi robots," *Journal of Electronics (China)*, vol. 29, pp. 541–549, 10 2012.

[45] R. Chandrasekar, R. Suresh, and S. 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*, pp. 628–629, IEEE, 2006.

[46] B. Wei, Q. Peng, Q. Zhang, and C. Li, "Identification of a combination of snps associated with graves' disease using swarm intelligence.," *Science China. Life sciences*, vol. 54, pp. 139–145, 2 2011.

[47] M. Afzalan and M. A.Taghikhani, "Dg placement and sizing in radial distribution network using pso&hbmo algorithms," *Energy and Power*, vol. 2, pp. 61–66, 8 2012.

[48] G. Anescu, "Particle swarm clustering optimization - a novel swarm intelligence approach to global optimization," *Annals of West University of Timisoara - Mathematics*, vol. 51, pp. 3–24, 1 2013.

[49] B. Karlik, "Soft computing methods in bioinformatics: A comprehensive review," *Mathematical and Computational Applications*, vol. 18, pp. 176–197, 12 2013.

[50] R. Bodkhe and D. Sain, "Improved the response throughput of load balancing of scientific cloud using particle of swarm optimization," *International Journal of Computer Applications*, vol. 143, pp. 26–30, 6 2016.

[51] J.-Y. Park and S. Y. Han, "Swarm intelligence topology optimization based on artificial bee colony algorithm," *International Journal of Precision Engineering and Manufacturing*, vol. 14, pp. 115–121, 12 2012.

[52] R. P. Ankala, D. Kavitha, and D. Haritha, "Mobile agent based routing in manets –attacks & defences," *Network Protocols and Algorithms*, vol. 3, pp. 108–121, 12 2011.

[53] C. Ramachandran, S. Misra, and M. Obaidat, "On evaluating some agent-based intrusion detection schemes in mobile ad-hoc networks," in *Proceedings of the SPECTS 2007*, (San Diego, CA), pp. 594–601, July 2007.

[54] J. Pugh and A. Martinoli, "Distributed scalable multi-robot learning using particle swarm optimization," *Swarm Intelligence*, vol. 3, pp. 203–222, 5 2009.

[55] G. Tuna, S. M. Potirakis, and G. Koulouras, "Implementing a trust and reputation model for robotic sensor networks," *Elektronika ir Elektrotechnika*, vol. 19, pp. 3–8, 12 2013.

[56] W. Deng, R. Chen, B. He, Y. Liu, L. Yin, and J. Guo, "A novel two-stage hybrid swarm intelligence optimization algorithm and application," *Soft Computing*, vol. 16, pp. 1707–1722, 5 2012.