Simulation of End-to-End Data Ingestion Strategies under Continuous Schema Drift

Salman Alutawa[1] and Hassan Alfardan[2]

[1]Department of Computer Science and Business, Bahrain National College of Technology, 12 Jaber Al-Ahmad Road, Tubli 706, Bahrain
[2]Department of Computer Science and Business, Al-Zahra University of Applied Sciences, 22 Sheikh Abdullah Avenue, Hidd 214, Bahrain

Abstract

Modern analytical platforms ingest data from heterogeneous and rapidly evolving sources, including microservices, third-party APIs, log streams, and event buses. In these environments, schemas rarely remain stable for long periods and tend to change incrementally as product features evolve, data producers refactor internal representations, or governance rules require new attributes. The resulting continuous schema drift challenges traditional end-to-end data ingestion pipelines that implicitly assume stable contracts between producers and consumers. When schemas drift while ingestion logic remains static, data loss, ingestion failures, and semantic inconsistencies can occur, especially for systems that enforce schemas eagerly and rely on manual change management processes. This work examines how different end-to-end ingestion strategies behave when exposed to continuous schema drift by constructing a simulation framework that models sources, pipelines, and storage targets over long horizons. The framework encodes schemas, transformations, and adaptation policies and replays drift scenarios with controllable frequency, magnitude, and correlation across sources. Several classes of ingestion strategy are considered, including schema-on-write with strict validation, schema-on-write with relaxed evolution, schema-on-read with late binding of structure, and hybrid patterns that separate physical and logical schemas. For each strategy, the simulation measures ingestion coverage, failure rates, lag between producer changes and consumer visibility, and resource consumption associated with reprocessing or schema migration. The study focuses on qualitative behavior and structural trade-offs rather than optimizing any single metric. The simulation results highlight how drift characteristics, such as directionality and concentration on specific fields, interact with pipeline design choices and suggest design considerations for constructing ingestion systems that remain operational under persistent schema evolution.

1 Introduction

End-to-end data ingestion pipelines connect data-producing components to analytical and operational consumers through a sequence of collection, transport, transformation, and storage stages [1]. In many organizations, these pipelines form the backbone of reporting, monitoring, and machine learning workflows. The correctness and timeliness of downstream decisions often depend on a tacit assumption that schemas describing the ingested data remain relatively stable or change in a carefully orchestrated manner. However, the shift toward distributed microservices, continuous deployment, and external data partnerships has weakened this assumption. Schemas now change frequently and asynchronously, driven by upstream product evolution rather than centrally coordinated data modeling processes [2].

Schema drift refers to the phenomenon where the schema associated with a data stream changes over time without a coordinated, global migration. Drift can be intentional, such as the addition of new fields for emerging use cases, or unintentional, such as type changes or semantic reinterpretations that are poorly documented. Continuous schema drift emphasizes that such changes are not isolated events but an ongoing property of the data ecosystem. Under continuous drift, even conservative ingestion strategies encounter a nontrivial number of schema-related incidents, and the combined effect over long horizons can degrade the reliability of downstream datasets.

Ingestion architectures exhibit a wide range of behaviors when exposed to schema drift [3]. Strategies that enforce strong contracts at the boundary, often referred to as schema-on-write, prioritize early validation and reject records that violate expected structure. This can preserve internal consistency but increases the likelihood of data loss when producers evolve more quickly than ingestion logic. In contrast, schema-on-read approaches defer schema

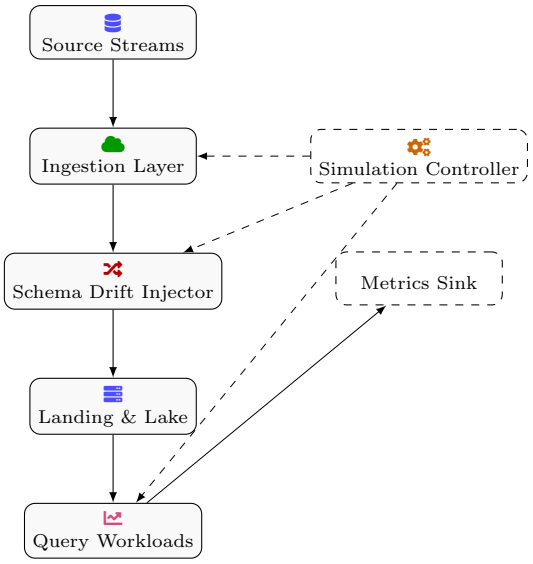

Figure 1: End-to-end simulation architecture with streams entering an ingestion layer under controlled schema drift, landing in a storage tier and being consumed by query workloads while a controller orchestrates runs and aggregates metrics.

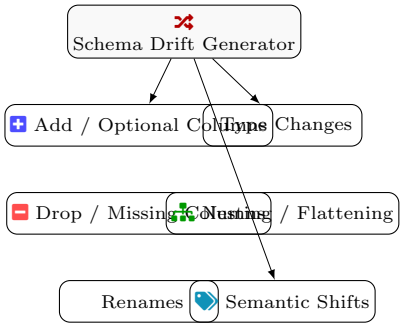

Figure 2: Schema drift generator decomposed into structural and semantic operations that can be scheduled over time to perturb incoming streams and emulate realistic evolution patterns.

binding to query or transformation time, enabling ingestion of semi-structured payloads but pushing the burden of managing drift onto consumers and query definitions. Between these poles lie hybrid patterns that attempt to combine the operational robustness of tolerant ingestion with the analytical convenience of well-structured schemas.

Empirically evaluating ingestion strategies under drift using live production systems is difficult [4]. Realistic drift patterns emerge over months or years, involve sensitive data, and are intertwined with organizational processes that are hard to replicate. Replaying historical logs with altered schemas provides partial insight but depends on the availability of archival data and does not allow controlled manipulation of drift parameters. These challenges motivate simulation-based approaches that treat drift as a stochastic process and ingestion strategies as policy functions operating on abstracted schemas and metrics.

This paper develops a simulation framework for examining end-to-end data ingestion under continuous schema drift. The framework models streams as sequences of records associated with time-varying schemas and represents ingestion pipelines as compositions of parameterized operators [5]. Drift processes generate schema changes according to configurable rules that control the rate of field additions, removals, and type modifications, as well as correlations across related streams. Ingestion strategies are encoded as adaptation policies that decide how and when to update internal schemas, whether to accept partially compatible records, and how to treat unknown or deprecated fields.

The analysis focuses on the interaction between drift patterns and ingestion policies, emphasizing qualitative insights about structural trade-offs rather than precise numerical predictions for any specific deployment. The simulation outputs metrics related to coverage, reliability, and cost across long time horizons, permitting comparison of strategies under shared drift scenarios. The discussion considers how these results might inform the design of robust ingestion architectures, while also acknowledging the simplifications inherent in the model and outlining directions for refining the representation of drift and adaptation dynamics [6].

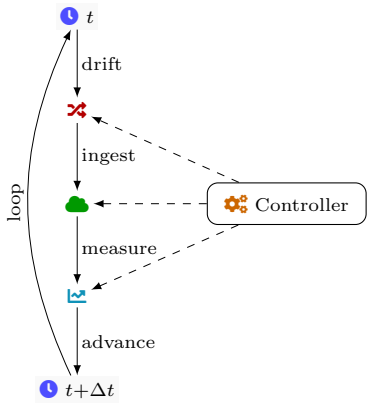

Figure 3: Discrete-time simulation loop in which each time step injects drift, executes ingestion, records metrics, and advances the clock, closed by a feedback loop driven by a configurable controller.

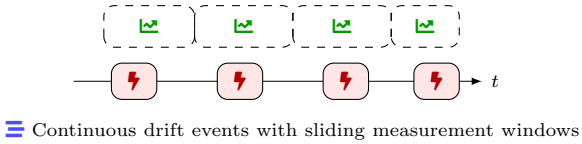

Figure 4: Timeline of continuous schema drift events with superimposed sliding windows, highlighting how short-lived and long-lived effects on ingestion quality are captured by the simulator.

2 Background on End-to-End Data Ingestion and Schema Drift

End-to-end data ingestion pipelines typically encompass several logical stages, even if they are not always implemented as distinct systems. At the boundary, connectors interface with external sources such as message queues, databases, or file drops, performing protocol translation and basic normalization. Incoming records are then transported through a messaging or streaming substrate that decouples producers from downstream consumers and provides durability guarantees. Transformation stages apply data cleaning, enrichment, and structural mapping before the data is written to storage systems such as data warehouses, data lakes, or specialized analytical stores. Downstream tasks, including reporting or model training, often consume data from curated layers derived from these storage systems [7].

Schemas appear in multiple forms along this path [8]. At the source, schemas may be explicit, as in column definitions for relational tables or strongly typed event definitions, or implicit, as in loosely structured JSON payloads. Within ingestion pipelines, schemas guide transformation logic, such as mapping fields to internal representations or enforcing type constraints. At storage, schemas define table structures or view definitions that enable efficient querying. In some architectures, schemas at different stages may diverge intentionally, for example when raw data is stored in semi-structured form while curated layers adopt normalized schemas tailored for analytics.

Schema drift arises when any of these schemas changes over time without synchronized updates across all dependent components [9]. Several basic forms of change include adding new fields, removing existing fields, altering types, splitting or merging fields, and changing value encodings or semantics without modifying field names. Drift can also involve restructuring nested data, such as moving attributes between levels in a hierarchy. While schema versioning practices aim to make such changes explicit, continuous delivery and decentralized ownership of services often lead to partial or delayed propagation of versioning information.

Continuous schema drift emphasizes the temporal dimension of these changes. Rather than treating schema evolution as a sequence of discrete, rare events, it views schemas as inherently time-indexed objects with ongoing modifications [10]. For example, a set of microservices may introduce new optional fields on a weekly basis, adjust ranges for numerical attributes in response to new business rules, or gradually migrate from one representation of identifiers to another. From the perspective of ingestion, this generates a stream of compatibility challenges that must be managed without assuming long periods of stability.

Different ingestion strategies embody different assumptions about how to handle schema drift. In strict schema-on-write systems, the ingestion pipeline maintains an authoritative description of acceptable schemas and validates each record at the point of entry. Records that do not conform are either rejected or quarantined [11]. This approach simplifies downstream processing because internal storages can assume that data matches expected structure. However, when source schemas drift more quickly than ingestion logic can be updated, strict validation can lead to sustained loss of records or increased operational overhead due to frequent schema migrations.

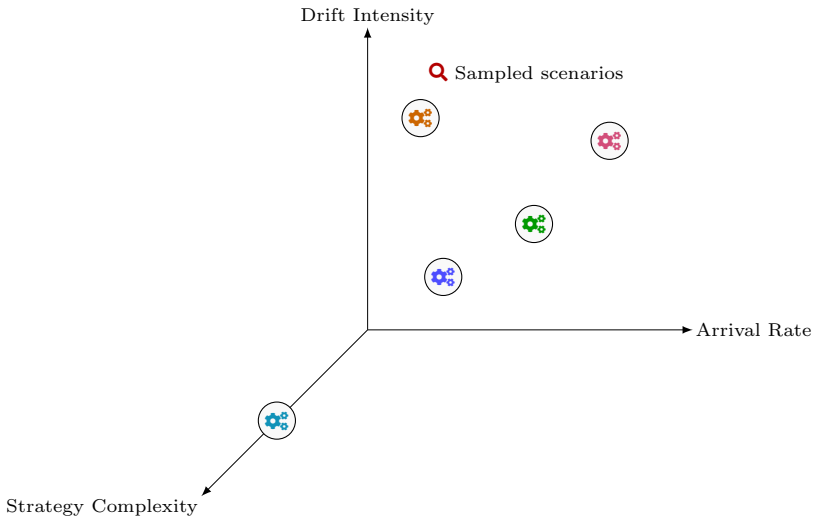

Figure 5: Configuration space spanned by drift intensity, event arrival rate, and ingestion strategy complexity, with sampled points representing concrete simulation runs used for systematic performance analysis.

Schema-on-read strategies invert this pattern by ingesting data in a more permissive form, often storing semi-structured payloads in formats that preserve unknown or variable fields. Schemas are applied at query time, allowing consumers to interpret the stored payload according to their current understanding [12]. This simplifies ingestion under drift but can complicate downstream logic, as consumers must handle evolving structures and potential inconsistencies. Hybrid designs may retain a relatively stable core schema with strict guarantees while allowing flexible extensions through auxiliary structures that are interpreted more dynamically.

Understanding the trade-offs between these strategies under continuous schema drift requires more than qualitative reasoning. Drift can interact with pipeline design in intricate ways, especially when multiple sources share related schemas or when fields have different importance levels for downstream tasks. For instance, frequent additions of optional fields may be benign for most strategies, while subtle type changes on core identifiers can trigger cascading failures [13]. A systematic examination must therefore consider how drift patterns, ingestion policies, and evaluation metrics combine to produce long-term behavior.

In this context, simulation offers a way to isolate and analyze the structural effects of schema drift on ingestion pipelines. By abstracting away from the details of specific technologies and focusing on the evolution of schemas, mappings, and policies over time, a simulation can illuminate patterns that are otherwise obscured by the complexity of real systems. The following sections formalize this abstraction and describe how linear models can be used to represent key aspects of drift and ingestion dynamics while keeping the framework tractable.

3 Simulation Framework and Linear Formalization

Table 1: Data ingestion strategies considered in the simulation study

Strategy	Description	Adaptation Mode	Update Granularity
Batch ETL	Periodic full loads with strict schemas	Offline replay	Daily / hourly
Micro-batch ETL	Small, frequent batches	Semi-online	Minutes
Streaming + Registry	Strong contracts via schema registry	Online negotiation	Per version
Streaming + Late Binding	Flexible schema-on-read	Online inference	Per query
Hybrid Pipeline	Combined batch + streaming path	Mixed	Multi-scale

Table 2: Taxonomy of schema drift types modelled in the simulator

Drift Type	Example	Structural Change	Breaking Risk
Additive field	Add `user_age`	Column addition	Low
Deprecated field	Drop `legacy_id`	Column removal	Medium
Renamed field	`cost` → `unit_price`	Column rename	High
Type change	`string` → `int`	Domain change	High
Nested restructuring	Flatten `address.street`	Path changes	High

Table 3: Key simulation parameters and explored value ranges

Parameter	Symbol	Values
Stream length (events)	N	$10^5, 10^6, 10^7$
Event rate (events/s)	λ	100, 1,000, 10,000
Drift rate (changes/day)	d	$1, 5, 10, 20$
Field cardinality	$\|F\|$	$20, 50, 100$
Burstiness (CV)	b	$0.5, 1.0, 2.0$

Table 4: Workload scenarios capturing different application domains

Scenario	Domain	Data Source	Characteristics
S1	E-commerce	Clickstream	High volume, additive drift
S2	Fintech	Transaction logs	Low loss tolerance, strict types
S3	IoT	Sensor telemetry	Bursty rates, nested records
S4	Marketing	Event tracking	Frequent renames, A/B tests
S5	Observability	Service metrics	Wide schemas, sparse fields

The simulation framework models a collection of data sources that emit records indexed by discrete time steps [14]. At each time step, every active source has an associated schema and an associated rate of record production. Schemas are treated as structured objects composed of fields, each with properties such as name, type, and an abstracted importance weight. Instead of tracking exact data values, the framework focuses on the interaction between schema evolution and ingestion policies, using aggregate metrics to approximate the volume and quality of ingested data.

Let the simulation operate over time steps indexed by integers $t = 1, 2, \ldots, T$. Consider N sources [15]. For each source i, a state vector $x_i(t)$ encodes the structural characteristics of its schema at time t. A simple representation partitions fields into categories, such as stable fields that have existed for many steps, newly added fields, and deprecated fields. The state can be modeled as a vector in $\mathbb{R}^d$, where entries count or weight fields in each category. For example, one may define

$$x_i(t) = \begin{pmatrix} s_i(t) \\ [16]n_i(t) \\ d_i(t) \end{pmatrix},$$

where $s_i(t)$ is the weighted count of stable fields, $n_i(t)$ is the weighted count of new fields introduced recently, and $d_i(t)$ is the weighted count of deprecated or soon-to-be-removed fields. The total number of fields is then approximated as

$$f_i(t) = s_i(t) + n_i(t) + d_i(t)[17].$$

Schema drift is represented as a stochastic process that evolves the state vectors over time. A linear approximation describes the expected evolution of $x_i(t)$ using a state transition matrix and a drift input. For each source i, one may write

$$x_i(t+1) = A_i x_i(t) + B_i u_i(t),$$

where A_i is a 3×3 matrix capturing deterministic tendencies such as the promotion of new fields to stable status and the aging of stable fields into deprecated status [18]. The vector $u_i(t)$ represents random drift inputs at time t, such as the arrival of brand new fields or abrupt deprecations, and B_i maps these into the state space. Although drift in real systems can be nonlinear and context dependent, a linear model provides a tractable baseline for exploring aggregate behavior.

The ingestion pipeline is modeled as a transformation that maps source states into ingestion outcomes. For each source i, define an ingestion state vector $z_i(t)$ that tracks how the pipeline currently interprets and stores fields from that source. This state includes counts of fields recognized by the pipeline, fields ignored, and fields mapped differently from their current source semantics [19]. A simple linear relationship can be used to approximate how ingestion decisions depend on source schemas and internal adaptation choice. Let

$$z_i(t+1) = C_i z_i(t) + D_i x_i(t),$$

where C_i encodes the inertia of the pipeline, such as the persistence of previously mapped fields, and D_i captures how new or deprecated fields influence the ingestion state.

Metrics used to evaluate ingestion performance are derived from these states. For example, the ingested coverage of source i at time t can be approximated as the ratio between the effective number of fields represented in the

Table 5: Evaluation metrics used to compare ingestion strategies

Metric	Definition	Range	Optimization Goal
Ingestion lag	Producer-to-sink delay	$[0, \infty)$ s	Minimize
Failure rate	Failed batches / window	$[0, 1]$	Minimize
Data loss	Dropped events	$[0, 1]$	Minimize
Manual intervention	Operator time per day	$[0, \infty)$ min	Minimize
Storage overhead	Extra bytes vs baseline	$[0, \infty)$ %	Minimize

Table 6: Aggregate performance under continuous schema drift (scenario S3, $d = 10$)

Strategy	Throughput (MB/s)	Data Loss (%)	Repair Time (s)
Batch ETL	45.2	7.8	1,240
Micro-batch ETL	51.6	4.3	610
Streaming + Registry	56.9	1.1	220
Streaming + Late Binding	53.4	0.8	95
Hybrid Pipeline	58.7	0.6	130

ingestion state and the effective number of fields in the source state [20]. If $g_i(t)$ denotes the weighted count of fields that are successfully mapped and stored in a usable form, the coverage can be written as

$$\gamma_i(t) = \frac{g_i(t)}{f_i(t)}.$$

Similarly, a simple model for ingestion failure rate can be defined by associating a probability of rejection or error with mismatches between source and ingestion states. Let $h_i(t)$ count the weighted mismatched fields or incompatible type changes. A linear approximation for failure probability might be

$$\phi_i(t) = \alpha_i h_i(t), [21]$$

where α_i is a scaling parameter chosen so that $\phi_i(t)$ remains between zero and one for typical values of $h_i(t)$.

To relate these structural quantities to end-to-end pipeline outcomes, the framework introduces a vector $y_i(t)$ of observable metrics for each source, such as expected number of successfully ingested records, expected number of failing records, and an abstracted resource cost. A linear observation model then expresses

$$y_i(t) = H_i x_i(t) + K_i z_i(t), [22]$$

where H_i and K_i summarize how source schema complexity and ingestion configuration jointly influence observable results. For example, the number of successful records could depend on both the volume of produced records, represented in an extended version of $x_i(t)$, and the coverage and failure rates embedded in $z_i(t)$.

Although the actual simulation need not compute continuous-valued expectations, these linear relations guide the design of discrete update rules. The simulator maintains integer or rational counts for fields and records but organizes them according to the linear model, ensuring that system behavior over many steps can be interpreted in terms of approximate linear dynamics. This view enables analytical reasoning about stability, steady-state behavior, and sensitivity to drift parameters, which complement direct numerical experimentation [23].

Finally, sources are not independent in many realistic environments. Shared patterns of schema evolution, such as coordinated introduction of new identifiers or common logging libraries, induce correlations across $x_i(t)$ for different i. The framework accommodates this by extending the state to a global vector $x(t)$ that concatenates all source states and by modeling drift input as a joint process. A global linear update of the form

$$x(t + 1) = Ax(t) + Bu(t)[24]$$

captures situations where drift affecting one source influences others. This global representation becomes important when evaluating ingestion strategies that share schema management components or must maintain consistent interpretations of fields across sources, as it allows the simulation to capture cross-source coupling in a controlled manner.

4 Ingestion Strategies and Linear Models of Adaptation

Ingestion strategies determine how the pipeline responds to schema drift and, consequently, how the ingestion state $z_i(t)$ evolves over time relative to the source state $x_i(t)$. These strategies can be expressed as adaptation

Table 7: Impact of schema evolution tooling on operator load

Configuration	Schema Discovery	Time to Recovery (s)	Operator Actions / day
C0: Baseline	Manual only	1,320	34.1
C1: + Drift Detector	Automatic	540	19.5
C2: + Change Recommender	Automatic	310	11.2
C3: + Canary Validation	Automatic	280	7.6

Table 8: Sensitivity of successful ingestion to schema drift rate

Drift Rate (changes/day)	Adaptive ETL (%)	Schema-on-Read (%)	Hybrid (%)
1	99.1	99.6	99.7
5	96.3	98.7	99.2
10	92.4	97.2	98.5
15	88.7	95.1	97.4
20	83.5	92.8	96.1

policies that map observed discrepancies between source schemas and current ingestion configurations into changes to internal schemas, transformation logic, and validation rules. While operational implementations may rely on complex rule engines or manual interventions, the simulation represents these policies using simplified linear or piecewise linear update functions [25].

A strict schema-on-write strategy maintains an internal schema for each source that it treats as authoritative. At each time step, the ingestion logic compares the current source schema to this internal schema. When a mismatch is detected, one of two actions is taken: either new records that do not conform are rejected, or the internal schema is updated to match the source after some delay that approximates manual engineering work. In the linear model, the internal schema can be encoded as a vector $w_i(t)$ with the same dimension as the source state. The discrepancy between source and ingestion expectations is then [26]

$$\delta_i(t) = x_i(t) - w_i(t),$$

and adaptation events can be modeled as updates of the form

$$w_i(t+1) = w_i(t) + L_i \delta_i(t),$$

where L_i is a matrix that controls how quickly and in which components the ingestion schema converges toward the source schema. A strict strategy might use a matrix that heavily weights stable fields and underweights new or deprecated fields, reflecting a preference for changes that affect only well-understood attributes [27].

The effect of these adaptations on the ingestion state $z_i(t)$ depends on how internal schema changes propagate to transformations and validation rules. For example, the number of recognized fields in $z_i(t)$ will increase when $w_i(t)$ is updated to include new fields, while the number of mismatched fields decreases when deprecated fields are removed. A linear model for this propagation can be written as

$$z_i(t+1) = M_i z_i(t) + N_i w_i(t+1),$$

where M_i captures persistence of previous ingestion decisions and N_i projects the updated schema into the ingestion state representation [28].

A schema-on-read strategy maintains a looser coupling between source schemas and ingestion logic. The pipeline accepts most records and stores them in a way that preserves unknown or variable structure, for instance through flexible payload columns or semi-structured formats. In this case, the ingestion schema $w_i(t)$ mainly constrains the minimal set of metadata needed to route and store records but does not strictly control the internal representation of fields. Instead, consumers define their own logical schemas when querying stored data. For the purposes of ingestion simulation, the main effect is that the tolerance to unknown fields is higher and the adaptation matrix L_i emphasizes preserving backward compatibility in stored payloads rather than aligning every detail with the latest source schema [29].

The linear representation of a schema-on-read strategy thus modifies the mapping from discrepancies $\delta_i(t)$ to ingestion state updates. Unknown fields increase the complexity of stored payloads but do not necessarily cause ingestion failures. This can be modeled by assigning smaller weights in the transformation from discrepancies to failure-related components of $z_i(t)$. One can express this schematically as

$$z_i(t+1) = M_i z_i(t) + N_i w_i(t) + P_i \delta_i(t), [30]$$

where the matrix P_i has small entries in rows corresponding to failure metrics but larger entries in rows representing storage cost or query complexity.

Hybrid strategies introduce additional structure by distinguishing between a stable core schema and a flexible extension region. The core schema is managed similarly to strict schema-on-write, with relatively conservative adaptation rules, while the extension region is treated more like schema-on-read, accepting new fields with minimal friction. In the linear model, this can be represented by splitting the state into core and extension components [31]. For each source i, write

$$x_i(t) = \begin{pmatrix} x_i^c(t) \\ x_i^e(t) \end{pmatrix}, \quad w_i(t) = \begin{pmatrix} w_i^c(t) \\ w_i^e(t) \end{pmatrix},$$

where superscripts c and e denote core and extension parts. Adaptation matrices are then block structured, with different response rates for core and extension discrepancies [32]. For instance,

$$w_i(t+1) = \begin{pmatrix} w_i^c(t+1) \\ w_i^e(t+1) \end{pmatrix} = \begin{pmatrix} w_i^c(t) \\ w_i^e(t) \end{pmatrix} + \begin{pmatrix} L_i^c & 0 \\ 0 & L_i^e \end{pmatrix} \begin{pmatrix} \delta_i^c(t) \\ \delta_i^e(t) \end{pmatrix},$$

with L_i^c chosen to be small in magnitude, modeling cautious updates for core fields, and L_i^e larger, modeling more aggressive acceptance of extension fields.

Across all strategies, the simulator evaluates a cost function that aggregates metrics such as data loss, operational changes, and resource usage. A simple quadratic cost over a time horizon can be defined as

$$J = \sum_{t=1}^{T} \sum_{i=1}^{N} \left([33] c_i^\top y_i(t) \right),$$

where c_i is a vector of weights encoding the relative importance of different metrics in $y_i(t)$. Although this cost is linear in $y_i(t)$, the dependence on adaptation parameters embedded in matrices such as L_i, M_i, and N_i can lead to complex trade-offs. For example, a larger adaptation rate in L_i may reduce data loss in the short term by aligning ingestion schemas more quickly with drifting sources but may increase operational instability if frequent schema changes propagate downstream [34].

The linear modeling perspective also supports comparative analysis of strategies by examining their stability properties. Consider the joint evolution of source and ingestion states under a fixed strategy. If the combined system can be approximated as

$$\begin{pmatrix} x(t+1) \\ z(t+1)[35] \end{pmatrix} = \begin{pmatrix} A & 0 \\ D & C \end{pmatrix} \begin{pmatrix} x(t) \\ z(t) \end{pmatrix} + \begin{pmatrix} B \\ E \end{pmatrix} u(t),$$

then the eigenvalues of the block matrix provide insight into whether discrepancies between source and ingestion schemas tend to converge or diverge over time for given drift characteristics. Strategies with adaptation parameters that lead to eigenvalues close to one may result in persistent misalignments under continuous drift, while more responsive configurations may drive the system toward a moving equilibrium where ingestion states track source changes with limited lag.

By framing ingestion strategies as parameterized linear adaptation policies, the simulation framework allows systematic exploration of how different design choices influence long-term behavior under continuous schema drift [36]. The next section describes how drift processes, strategy parameters, and evaluation metrics are instantiated in numerical experiments that compare strategy families across a range of drift scenarios.

5 Experimental Design and Simulation Results

The experimental component of the study uses the simulation framework to compare ingestion strategies under controlled schema drift scenarios. The simulation environment instantiates a set of synthetic sources, each with an initial schema and an associated production intensity representing the relative volume of records. Drift processes are configured to control the rate at which fields are added, removed, or modified and to induce different patterns of correlation across sources. The objective is not to replicate any single real system but to explore how structural parameters of drift interact with ingestion strategies in a simplified but interpretable setting [37].

At the beginning of each simulation run, every source is assigned an initial state vector $x_i(1)$ with nonzero values for stable fields and zero for new and deprecated fields. The drift process is specified by choosing a transition matrix A_i and an input matrix B_i for each source, along with a distribution for the drift input $u_i(t)$. To model continuous schema drift with a moderate rate of change, the diagonal entries of A_i associated with stable fields are chosen close to one, while off-diagonal entries promote a small flow of mass from new to stable and from stable to deprecated categories. The drift input injects new fields with a probability proportional to production intensity, representing the idea that more active sources are more likely to evolve.

Different drift regimes are explored by varying the magnitude and structure of the input distribution [38]. In a low-drift regime, the expected number of new fields introduced at each step is small relative to the existing

number of stable fields, and deprecations occur infrequently. In a high-drift regime, new fields are introduced at a rate comparable to the size of the existing schema, and fields transition more quickly from new to deprecated. A correlated drift regime introduces dependencies across sources by defining $u(t)$ as a low-dimensional random vector that is mapped into each source through the matrix B, resulting in simultaneous introductions or changes affecting multiple sources.

Each ingestion strategy is parameterized by adaptation matrices such as L_i, M_i, and N_i, which control the responsiveness of the pipeline to schema discrepancies. For strict schema-on-write, the simulation sets small adaptation rates, corresponding to teams that change ingestion schemas only after accumulating substantial discrepancies or after failures are observed [39]. For schema-on-read, higher adaptation rates are used for components that capture logical views but lower rates for components that influence validation, reflecting the higher tolerance for unknown fields at ingestion time. Hybrid strategies are configured by assigning different adaptation rates to core and extension components, with core fields adapting conservatively and extension fields more aggressively.

The numerical experiments track several metrics over the time horizon. For each source and time step, the simulator computes approximate coverage $\gamma_i(t)$, failure probability $\phi_i(t)$, and a normalized cost representing additional storage or processing overhead. These per-source metrics are then aggregated into system-wide summaries, such as average coverage across sources, maximum failure probability, and cumulative cost [40]. The aggregate coverage at time t can be expressed as

$$\Gamma(t) = \frac{1}{N} \sum_{i=1}^{N} \gamma_i(t),$$

and the aggregate failure measure as

$$\Phi(t) = \frac{1}{N} \sum_{i=1}^{N} \phi_i(t).$$

Cumulative cost over the horizon is computed by summing a weighted combination of observation vectors as previously described.

In the low-drift regime, all strategies maintain relatively high coverage throughout the simulation, but they differ in how they handle occasional bursts of schema change [41]. Strict schema-on-write strategies exhibit temporary drops in coverage and spikes in failure probability when a drift event introduces several new fields simultaneously or modifies key fields. These effects persist for several time steps until adaptation updates align ingestion schemas with sources. In contrast, schema-on-read strategies absorb most changes without immediate failures, sustaining coverage closer to the theoretical maximum imposed by the drift process. However, they accumulate higher storage and complexity costs because new fields are ingested even when they are not yet relevant for downstream consumers.

Hybrid strategies in the low-drift regime behave in between these extremes [42]. Core fields, which represent structurally important attributes, show behavior similar to strict schema-on-write, with occasional coverage dips during major changes. Extension fields are handled more flexibly, and their coverage and failure patterns resemble schema-on-read. The net effect is that overall coverage stays high, while failure incidents on critical fields remain rare. Cumulative cost sits between the two extremes, as hybrid strategies ingest many but not all extension fields.

In the high-drift regime, differences between strategies become more pronounced [43]. Strict schema-on-write strategies struggle to keep up with continual changes, and average coverage declines as the constant influx of new fields creates persistent discrepancies. Failure probabilities increase, leading to a sustained fraction of records being rejected or delayed. Frequent schema migrations in the ingestion pipeline also raise the operational cost component of the simulation, even though this cost is represented abstractly. Schema-on-read strategies, by design, continue to ingest most records, so coverage remains high, but the complexity of stored payloads increases steadily. The linear model captures this through growth in components of $z_i(t)$ that represent unknown or rarely used fields [44].

Hybrid strategies show varying behavior depending on how aggressively the extension part adapts. Aggressive extensions behave similarly to schema-on-read, maintaining high coverage but incurring significant complexity cost. Conservative extensions reduce complexity but may miss fields that quickly become important for downstream tasks. The simulation indicates that there is a band of intermediate adaptation rates that yields balanced behavior, with coverage that remains high for core and frequently used extension fields while limiting the accumulation of seldom used attributes.

Correlated drift introduces additional considerations [45]. When multiple sources evolve in similar ways, for instance by introducing a shared identifier or logging pattern, ingestion strategies that share schema management components can capitalize on this structure. In the linear model, shared adaptation is represented by coupling terms in matrices such as L_i, allowing a change detected in one source to influence ingestion schemas for others. Simulations show that under correlated drift, such shared adaptation can reduce coverage gaps and failure spikes across sources at the cost of more complex coordination logic. Schema-on-read strategies benefit less from this effect at ingestion time but may exploit shared structure when defining downstream logical schemas.

Overall, the simulation results highlight that no single strategy dominates across all drift regimes and evaluation metrics [46]. Strict schema-on-write offers strong guarantees when drift is slow and well-managed but becomes

brittle under high, unpredictable drift. Schema-on-read provides robustness to change but shifts complexity to storage and query layers, which may be acceptable or problematic depending on context. Hybrid strategies offer a range of trade-offs that can be tuned through adaptation parameters to align with organizational tolerance for data loss, operational overhead, and analytical complexity. These findings are qualitative rather than prescriptive, reflecting the abstract nature of the simulation, but they illustrate how a linear modeling framework coupled with controlled experiments can illuminate the behavior of ingestion strategies under continuous schema drift.

6 Discussion

The simulation-based analysis provides a structured lens for examining how ingestion strategies behave under continuous schema drift, but its abstractions and assumptions merit careful interpretation [47]. The linear models for schema evolution and ingestion adaptation, while deliberately simplified, capture several aspects of real systems that are otherwise difficult to reason about analytically. For example, the state vectors representing stable, new, and deprecated fields provide a coarse but useful distinction among parts of the schema that tend to exhibit different drift patterns and different importance for downstream consumers. The separation of source state, ingestion state, and observation metrics clarifies the chain of influence from producer changes to observable ingestion outcomes.

One insight from the experiments is the importance of adaptation rates in mediating the trade-off between responsiveness and stability. In the linear framework, these rates appear as entries in matrices such as L_i and M_i, which determine how quickly ingestion schemas and logic adjust to discrepancies [48]. Very small adaptation rates correspond to pipelines that require significant human intervention or governance processes before adopting schema changes. Such pipelines may perform well when drift is rare and predictable but experience prolonged periods of reduced coverage under persistent drift. In contrast, large adaptation rates approximate highly automated systems that quickly mirror source changes, limiting coverage gaps but potentially propagating transient or undesirable schema modifications downstream. The simulation makes this trade-off explicit by linking adaptation parameters to long-term metrics such as average coverage and cumulative cost.

The comparative behavior of schema-on-write and schema-on-read strategies in different drift regimes highlights another structural consideration [49]. Schema-on-write can be seen as embedding a projection operator that maps the high-dimensional space of possible source schemas into a lower-dimensional space of accepted schemas. When drift trajectories frequently leave this subspace, the projection leads to rejection or truncation of information. Schema-on-read maintains a higher-dimensional representation at ingestion time, deferring projection to query time. The simulation implicitly represents this by assigning different weights to unknown fields in the ingestion state and by modeling different relationships between discrepancies and failure probabilities [50]. The resulting behavior underscores that the choice between early and late binding of schema is closely tied to expectations about the volume and direction of drift.

Hybrid strategies offer a mechanism for integrating these perspectives by assigning different binding policies to different parts of the schema. The linear decomposition into core and extension components emphasizes that not all fields are equally sensitive to drift. Core fields, such as identifiers and key metrics, often require stricter control because they underpin primary analytical and operational functions. Extension fields, such as auxiliary attributes or experimental signals, can tolerate more flexible handling [51]. By tuning adaptation rates and cost weights separately for these components, the simulation can explore how different allocations of flexibility influence aggregate behavior. The experiments suggest that hybrid configurations can achieve combinations of coverage and complexity that are not easily obtained with uniform strategies.

Despite these conceptual benefits, several limitations constrain the scope of the simulation. The linear models do not capture nonlinear phenomena such as thresholds in organizational response, where a sudden increase in failures triggers dedicated engineering efforts that dramatically alter adaptation rates. They also abstract away semantic aspects of schema drift, such as subtle changes in meaning that are not reflected in structural attributes [52]. In reality, a field may retain the same name and type while its interpretation shifts due to a change in upstream logic, causing silent changes in derived metrics that the simulation, focused on structural states, does not represent.

Moreover, the cost models used in the experiments are necessarily simplified. Operational costs associated with schema migration involve coordination across teams, testing, and deployment, which may not scale linearly with the magnitude of schema changes. Storage and query complexity costs depend on access patterns, indices, and query optimizations that are highly system dependent. The simulation treats these costs as linear functions of counts of fields and discrepancies, which supports comparative analysis but cannot predict actual resource usage in specific deployments [53]. Consequently, the results should be interpreted as indicative of relative trends rather than quantitative forecasts.

Another consideration is that continuous schema drift is not purely stochastic but often driven by organizational and product dynamics. For instance, periods of rapid product innovation may correspond to high drift, followed by stabilization phases. Some changes are coordinated across services, while others arise from local decisions. The drift

processes in the simulation approximate these patterns through parameters controlling drift rate and correlation but do not incorporate explicit models of organizational decision making [54]. A richer framework could integrate models of change management practices, release cadences, and governance policies, allowing for more nuanced scenarios where drift is influenced by feedback from ingestion performance.

Finally, the simulation focuses on ingestion behavior and does not model downstream adaptation in detail. In reality, consumers of ingested data, such as analytics and machine learning pipelines, also adapt to schema drift through changes in queries, feature definitions, and model architectures. The interplay between ingestion adaptation and downstream adaptation can shape overall system resilience. For example, a schema-on-read ingestion strategy may rely on downstream teams to maintain robust query logic that handles evolving structures [55]. If downstream adaptation is slow or inconsistent, the apparent robustness at ingestion may translate into fragility at the consumption layer. Extending the simulation to include simple models of downstream adaptation would allow exploration of end-to-end behavior beyond ingestion.

These limitations do not negate the utility of the current framework but indicate directions for refinement. Future work could relax linearity assumptions for specific components, incorporate event-driven adaptation mechanisms that respond nonlinearly to observed failures, and couple ingestion dynamics to models of downstream processing. Nevertheless, even in its current form, the simulation clarifies how continuous schema drift interacts with end-to-end ingestion strategies and highlights that pipeline design decisions implicitly encode assumptions about drift patterns and adaptation capacities [56].

7 Conclusion

This study has examined the behavior of end-to-end data ingestion strategies under continuous schema drift through a simulation framework that combines abstract representations of schemas, ingestion states, and adaptation policies with linear models of their evolution. By expressing source schemas as time-indexed state vectors and modeling drift as a stochastic process governed by transition and input matrices, the framework captures how structural changes in data sources accumulate over time. Ingestion strategies are represented as parameterized adaptation policies that map discrepancies between source and internal schemas into updates of ingestion configuration, enabling systematic comparison of schema-on-write, schema-on-read, and hybrid approaches.

The simulation experiments demonstrate that the interaction between drift characteristics and ingestion strategy parameters significantly influences long-term metrics such as coverage, failure rates, and complexity-related costs. Under low drift, strict schema-on-write strategies can maintain high coverage with limited operational overhead, while schema-on-read and hybrid strategies provide additional flexibility for handling occasional changes [57]. Under high drift, strict schema-on-write becomes increasingly brittle, with persistent coverage gaps and elevated failure rates, whereas schema-on-read maintains higher coverage at the expense of increased storage and query complexity. Hybrid strategies offer intermediate behavior whose properties depend on how flexibility is allocated between core and extension components and on the chosen adaptation rates.

The linear modeling perspective, although simplified, offers a compact way to reason about these trade-offs and to explore how changes in strategy parameters affect system behavior across different drift regimes. It highlights the role of adaptation rates in balancing responsiveness to change against stability of downstream schemas and emphasizes that the choice between early and late binding of structure is not absolute but can vary within a schema. The analysis suggests that ingestion architectures benefit from making these design choices explicit, aligning them with expectations about drift intensity, governance capacity, and downstream tolerance for structural variability [58].

Future extensions of the framework could incorporate richer models of schema semantics, nonlinear adaptation mechanisms, and explicit representations of downstream consumers, allowing for more comprehensive end-to-end evaluations. They might also integrate organizational factors such as release cadences and change management practices that influence both drift processes and adaptation responses. While such extensions would increase model complexity, they could provide additional insight into how technical and organizational design decisions jointly shape the resilience of data ecosystems to continuous schema drift. the simulation approach presented here offers a structured means for exploring the dynamics of data ingestion strategies in environments where schemas evolve continually. It does not prescribe a single optimal strategy but provides a basis for understanding how different design choices behave under varying drift conditions, helping practitioners reason about the trade-offs inherent in building ingestion systems that remain functional as their data sources change over time [59].

References

[1] C. Zuo, L. Wu, Z.-F. Zeng, and H.-L. Wei, "Stochastic fractal based multiobjective fruit fly optimization," *International Journal of Applied Mathematics and Computer Science*, vol. 27, no. 2, pp. 417–433, Jun. 27, 2017. DOI: 10.1515/amcs-2017-0029

[2] P. Nagchoudhury and K. Choudhary, "Classification of swarm intelligence based clustering methods," *International Journal of Computer Applications*, vol. 91, no. 6, pp. 28–33, Apr. 18, 2014. DOI: 10.5120/15887-5078

[3] G. Sun, Y. Liu, H. Li, J. Li, A. Wang, and Y. Zhang, "Power-pattern synthesis for energy beamforming in wireless power transmission," *Neural Computing and Applications*, vol. 30, no. 7, pp. 2327–2342, Oct. 30, 2017. DOI: 10.1007/s00521-017-3255-6

[4] T. Srinivasan, V. Vijaykumar, and R. Chandrasekar, "An auction based task allocation scheme for power-aware intrusion detection in wireless ad-hoc networks," in *2006 IFIP International Conference on Wireless and Optical Communications Networks*, IEEE, 2006, 5–pp.

[5] J. Wang, B. Gong, H. Liu, and S. Li, "Multidisciplinary approaches to artificial swarm intelligence for heterogeneous computing and cloud scheduling," *Applied Intelligence*, vol. 43, no. 3, pp. 662–675, May 16, 2015. DOI: 10.1007/s10489-015-0676-8

[6] W. Mao, H.-y. Lan, and H.-r. Li, "A new modified artificial bee colony algorithm with exponential function adaptive steps," *Computational intelligence and neuroscience*, vol. 2016, no. 2016, pp. 9 820 294–9 820 294, May 17, 2016. DOI: 10.1155/2016/9820294

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

[8] W. Kramper, R. Wanker, and K.-H. Zimmermann, "Analysis of swarm behavior using compound eye and neural network control," *Open Computer Science*, vol. 2, no. 1, pp. 16–32, Mar. 1, 2012. DOI: 10.2478/s13537-012-0004-x

[9] M. Jafari and H. Khotanlou, "A routing algorithm based on ant colony, local search and fuzzy inference to improve energy consumption in wireless sensor networks," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 3, no. 5, pp. 640–650, Oct. 1, 2013. DOI: 10.11591/ijece.v3i5.3628

[10] A. Prorok, N. Correll, and A. Martinoli, "Multi-level spatial modeling for stochastic distributed robotic systems," *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 574–589, Apr. 1, 2011. DOI: 10.1177/0278364910399521

[11] R. S. Rashmi, R. Pratyusha, T. Haidar, S. Sujata, k. Balabantaray Bunil, and M. Sharmilla, "Navigational path planning of multi-robot using honey bee mating optimization algorithm (hbmo)," *International Journal of Computer Applications*, vol. 27, no. 11, pp. 1–8, Aug. 31, 2011. DOI: 10.5120/3348-4617

[12] Y. Yun and Y. Li, "An improved nonlinear multi-objective optimization problem based on genetic algorithm," *International Journal of Hybrid Information Technology*, vol. 9, no. 7, pp. 361–372, Jul. 31, 2016. DOI: 10.14257/ijhit.2016.9.7.33

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

[14] S. Xu et al., "Ant-based swarm algorithm for charging coordination of electric vehicles," *International Journal of Distributed Sensor Networks*, vol. 9, no. 5, pp. 268 942–, May 1, 2013. DOI: 10.1155/2013/268942

[15] H.-S. Park and N.-H. Tran, "A smart machining system," *Journal of the Korean Society for Precision Engineering*, vol. 32, no. 1, pp. 39–47, Jan. 1, 2015. DOI: 10.7736/kspe.2015.32.1.39

[16] S. A. Chalack, S. N. Razavi, and A. Harounabadi, "Job scheduling on the grid environment using max-min firefly algorithm," *International Journal of Computer Applications Technology and Research*, vol. 3, no. 1, pp. 63–67, Jan. 10, 2014. DOI: 10.7753/ijcatr0301.1014

[17] A. Lazarowska, "Swarm intelligence approach to safe ship control," *Polish Maritime Research*, vol. 22, no. 4, pp. 34–40, Dec. 1, 2015. DOI: 10.1515/pomr-2015-0068

[18] M. Sumathi, "Swarm intelligence based bee inspired routing protocol for multipath routing in manet," *IJARCCE*, pp. 204–208, Feb. 28, 2015. DOI: 10.17148/ijarcce.2015.4245

[19] T. Schmickl and K. Crailsheim, "Trophallaxis within a robotic swarm: Bio-inspired communication among robots in a swarm," *Autonomous Robots*, vol. 25, no. 1, pp. 171–188, Dec. 22, 2007. DOI: 10.1007/s10514-007-9073-4

[20] M. Arsuaga-Ríos and M. A. Vega-Rodríguez, "Multi-objective energy optimization in grid systems from a brain storming strategy," *Soft Computing*, vol. 19, no. 11, pp. 3159–3172, Oct. 5, 2014. DOI: 10.1007/s00500-014-1474-7

[21] S.-g. Jung, B. Kang, S. Yeoum, and H. Choo, "Trail-using ant behavior based energy-efficient routing protocol in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 12, no. 4, pp. 7 350 427–, Apr. 1, 2016. DOI: 10.1155/2016/7350427

[22] T. Srinivasan, R. Chandrasekar, V. Vijaykumar, V. Mahadevan, A. Meyyappan, and M. Nivedita, "Exploring the synergism of a multiple auction-based task allocation scheme for power-aware intrusion detection in wireless ad-hoc networks," in *2006 10th IEEE Singapore International Conference on Communication Systems*, IEEE, 2006, pp. 1–5.

[23] R.-H. Hwang and C.-C. Hoh, "Cross-layer design of p2p file sharing over mobile ad hoc networks," *Telecommunication Systems*, vol. 42, no. 1, pp. 47–61, Jun. 12, 2009. DOI: 10.1007/s11235-009-9168-7

[24] J. R. Srivastava and T. Sudarshan, "Energy-efficient cache node placement using genetic algorithm in wireless sensor networks," *Soft Computing*, vol. 19, no. 11, pp. 3145–3158, Oct. 8, 2014. DOI: 10.1007/s00500-014-1473-8

[25] K. S. Reddy, L. K. Panwar, R. Kumar, and B. K. Panigrahi, "Distributed resource scheduling in smart grid with electric vehicle deployment using fireworks algorithm," *Journal of Modern Power Systems and Clean Energy*, vol. 4, no. 2, pp. 188–199, Mar. 25, 2016. DOI: 10.1007/s40565-016-0195-6

[26] V. Balaji and V. Duraisamy, "Ant optimized link quality for ad hoc on demand distance vector," *Wireless Personal Communications*, vol. 79, no. 1, pp. 763–771, Jun. 21, 2014. DOI: 10.1007/s11277-014-1885-x

[27] J. Ju, W. Bao, Z. Wang, Y. Wang, and W. Li, "Research for the task scheduling algorithm optimization based on hybrid pso and aco for cloud computing," *International Journal of Grid and Distributed Computing*, vol. 7, no. 5, pp. 87–96, Oct. 31, 2014. DOI: 10.14257/ijgdc.2014.7.5.08

[28] A. Forestiero, "Bio-inspired algorithm for outliers detection," *Multimedia Tools and Applications*, vol. 76, no. 24, pp. 25 659–25 677, Feb. 4, 2017. DOI: 10.1007/s11042-017-4443-1

[29] M. Mastali, A. Kheyroddin, B. Samali, and R. Vahdani, "Optimal placement of active braces by using pso algorithm in near- and far-field earthquakes," *International Journal of Advanced Structural Engineering*, vol. 8, no. 1, pp. 29–44, Feb. 15, 2016. DOI: 10.1007/s40091-016-0111-3

[30] 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*, IEEE, 2006, pp. 628–629.

[31] M. Darbari and P. Sahai, "Adaptive e-learning multi-agent systems with swarm intelligence," *International Journal of Applied Information Systems*, vol. 7, no. 3, pp. 16–20, May 2, 2014. DOI: 10.5120/ijais14-451164

[32] S. Prajapat, A. Sharma, and R. S. Thakur, "Avk based cryptosystem and recent directions towards cryptanalysis," *Journal of Internet Computing and Services*, vol. 17, no. 5, pp. 97–110, Oct. 31, 2016. DOI: 10.7472/jksii.2016.17.5.97

[33] A. Kulkarni and D. Parle, "Multi objective particle swarm optimization," *International Journal of Engineering Research and*, vol. 6, no. 03, Mar. 18, 2017. DOI: 10.17577/ijertv6is030247

[34] V. Trianni, E. Tuci, and K. M. Passino, "Special issue on swarm cognition," *Swarm Intelligence*, vol. 5, no. 1, pp. 1–2, Dec. 23, 2010. DOI: 10.1007/s11721-010-0052-6

[35] I. Susnea, "The inn at the crossroads – a model of distributed spatial knowledge," *Studies in Informatics and Control*, vol. 25, no. 1, Mar. 5, 2016. DOI: 10.24846/v25i1y201602

[36] M. Fathian and B. Amiri, "A honeybee-mating approach for cluster analysis," *The International Journal of Advanced Manufacturing Technology*, vol. 38, no. 7-8, pp. 809–821, Aug. 14, 2007. DOI: 10.1007/s00170-007-1132-7

[37] K. M. bin Abdl and S. Z. M. Hashim, "Swarm-based feature selection for handwriting identification," *Journal of Computer Science*, vol. 6, no. 1, pp. 80–86, Jan. 1, 2010. DOI: 10.3844/jcssp.2010.80.86

[38] D. N. Kozlov, "Gathering identical autonomous systems on a circle using stigmergy," *Distributed Computing*, vol. 25, no. 6, pp. 461–472, Sep. 1, 2012. DOI: 10.1007/s00446-012-0178-4

[39] R. Chandrasekar and S. Misra, "Introducing an aco based paradigm for detecting wildfires using wireless sensor networks," in *2006 International Symposium on Ad Hoc and Ubiquitous Computing*, IEEE, 2006, pp. 112–117.

[40] L. Pagliarini and H. H. Lund, "An educational tool for interactive parallel and distributed processing," *Artificial Life and Robotics*, vol. 16, no. 4, pp. 441–447, Feb. 22, 2012. DOI: 10.1007/s10015-011-0996-7

[41] M. Siddappa and C. raju, "Survey on an efficient coverage and connectivity of wireless sensor networks using intelligent algorithms.," *International Journal of Information Technology and Computer Science*, vol. 4, no. 5, pp. 39–45, May 2, 2012. DOI: 10.5815/ijitcs.2012.05.06

[42] D. Li, Q. He, L. Chunli, and Y. Hongjie, "Local binary fitting segmentation by cooperative quantum particle optimization," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 15, no. 1, pp. 531–539, Mar. 1, 2017. DOI: 10.12928/telkomnika.v15i1.3159

[43] O. Deepa and A. Senthilkumar, "Swarm intelligence from natural to artificial systems: Ant colony optimization," *International Journal on Applications of Graph Theory In wireless Ad Hoc Networks And sensor Networks*, vol. 8, no. 1, pp. 9–17, Mar. 30, 2016. DOI: 10.5121/jgraphoc.2016.8102

[44] H. Hamann et al., "Hybrid societies: Challenges and perspectives in the design of collective behavior in self-organizing systems," *Frontiers in Robotics and AI*, vol. 3, no. 14, pp. 14–, Apr. 11, 2016. DOI: 10.3389/frobt.2016.00014

[45] R. Kalpana and N. Rengarajan, "Mobile anonymous trust based routing using ant colony optimization," *American Journal of Applied Sciences*, vol. 9, no. 8, pp. 1283–1289, Aug. 1, 2012. DOI: 10.3844/ajassp.2012.1283.1289

[46] I. Füzesi and M. Herdon, "It applications in the quality management of hungarian meat product chains," *Journal of Agricultural Informatics*, vol. 1, no. 1, Mar. 21, 2011. DOI: 10.17700/jai.2010.1.1.8

[47] 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*, IEEE, 2006, pp. 1–4.

[48] R. Quitadamo and F. Zambonelli, "Autonomic communication services: A new challenge for software agents," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 3, pp. 457–475, May 14, 2008. DOI: 10.1007/s10458-008-9054-9

[49] A. Kaveh, M. Ghafari, and Y. Gholipour, "Optimal seismic design of 3d steel moment frames: Different ductility types," *Structural and Multidisciplinary Optimization*, vol. 56, no. 6, pp. 1353–1368, Jun. 13, 2017. DOI: 10.1007/s00158-017-1727-z

[50] S. Saba, F. Ahsan, and S. Mohsin, "Bat-ann based earthquake prediction for pakistan region," *Soft Computing*, vol. 21, no. 19, pp. 5805–5813, May 10, 2016. DOI: 10.1007/s00500-016-2158-2

[51] J. Peña, C. A. Rouff, M. Hinchey, and A. Ruiz-Cortés, "Modeling nasa swarm-based systems: Using agent-oriented software engineering and formal methods," *Software & Systems Modeling*, vol. 10, no. 1, pp. 55–62, Oct. 9, 2009. DOI: 10.1007/s10270-009-0135-2

[52] X. Chen and T. Wang, "Threat assessment of aerial target based on modified pso optimized bp neural network," *International Journal of Control and Automation*, vol. 10, no. 2, pp. 103–114, Feb. 28, 2017. DOI: 10.14257/ijca.2017.10.2.09

[53] D.-p. Qu, X.-w. Wang, and M. Huang, "Component based ant routing protocols analysis over mobile ad hoc networks," *Journal of Central South University*, vol. 20, no. 9, pp. 2378–2387, Sep. 6, 2013. DOI: 10.1007/s11771-013-1747-9

[54] Y. Li and J. Pan, "Research of dynamic access to cloud database based on improved pheromone algorithm," *International Journal of Database Theory and Application*, vol. 9, no. 5, pp. 197–202, May 31, 2016. DOI: 10.14257/ijdta.2016.9.5.20

[55] T. A. Jilani, B. S.M.A., U. Amjad, and T. A. Siddiqui, "A particle swarm intelligence based fuzzy time series forecasting model," *International Journal of Computer Applications*, vol. 38, no. 10, pp. 47–52, Jan. 28, 2012. DOI: 10.5120/4742-6776

[56] 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*, 2006, pp. 1–6. DOI: 10.1109/ICCIS.2006.252326

[57] J. Gou, M.-Z. Chen, W. Luo, and F. Hou, "Programming learning style diagnosis scheme using pso-based fuzzy knowledge fusion," *Cybernetics and Information Technologies*, vol. 14, no. 1, pp. 84–100, Mar. 1, 2014. DOI: 10.2478/cait-2014-0007

[58] A. Jain and N. S. Chaudhari, "A novel cuckoo search strategy for automated cryptanalysis: A case study on the reduced complex knapsack cryptosystem," *International Journal of System Assurance Engineering and Management*, vol. 9, no. 4, pp. 942–961, Nov. 21, 2017. DOI: 10.1007/s13198-017-0690-9

[59] S. Zhang, C. K. M. Lee, K. M. Yu, and H. C. W. Lau, "Design and development of a unified framework towards swarm intelligence," *Artificial Intelligence Review*, vol. 47, no. 2, pp. 253–277, Apr. 26, 2016. DOI: 10.1007/s10462-016-9481-y