Peer reviewed Journals
seron press

RESEARCH

Unified Optimization Frameworks for Joint Retrieval, Clustering, and Classification in Large-Scale Information Systems

Esteban Cárdenas[1] and Mauricio Villalba[2]

[1]Universidad de Pamplona, Avenida Los Almeces 301, Department of Intelligent Computing, Pamplona, Norte de Santander, Colombia
[2]Universidad del Cauca, Carrera, Department of Advanced Information Engineering, Popayán, Cauca, Colombia

Abstract

In operational settings, learned vector representations functions are rarely isolated: retrieval quality depends on semantic structure that is also useful for clustering, while classification objectives shape representation geometry in ways that can either help or harm nearest-neighbor search. This paper develops a unified optimization view of joint retrieval, clustering, and classification, treating them as coupled objectives acting on shared embeddings, shared prototypes, and shared normalization constraints. The framework models retrieval as a metric-consistent ranking risk, clustering as a structured assignment of embeddings to latent groups, and classification as a supervised risk over the same embedding space, with explicit mechanisms for partial labels, class imbalance, and continuously arriving data. A central contribution is an integrated objective with controllable coupling strengths that admits scalable training via stochastic gradient steps, block-coordinate updates, and constrained assignment solvers. The paper analyzes how coupling induces trade-offs in embedding anisotropy, prototype collapse, and calibration of decision boundaries, and it proposes stability conditions that keep retrieval neighborhoods consistent under clustering and classification updates. Practical system considerations are addressed, including approximate nearest-neighbor indexing, delayed index refresh, streaming updates, and distributed training. The resulting framework provides a principled route for designing joint learners that match the intertwined requirements of modern retrieval-centric information systems.

1. Introduction

A user query is mapped to a representation, candidate items are retrieved by similarity search, and the retrieved set is further processed by classification or re-ranking models. In parallel, operators often need clustering to support [1] navigation, discovery, moderation, deduplication, corpus monitoring, and taxonomy construction. Although these functions can be built as separate pipelines, separation creates inconsistencies. A representation trained only for classification may form decision regions that are sufficient for top-1 accuracy yet poorly aligned with neighborhood structure required for retrieval. A representation trained only for retrieval may preserve fine-grained similarity but fail to produce stable clusters that operators can interpret or maintain. A representation trained only for [2] clustering may organize the corpus into coherent groups while ignoring labels that encode downstream policy or product requirements [3] [4].

A common response is to stage training: learn embeddings for retrieval, freeze them, cluster offline, then train a classifier on top. Staging can work when distributions are stable and objectives are weakly coupled. However, large-scale systems are rarely stable. Corpora drift due to new content and user behavior, label coverage changes over time, and retrieval indices must be refreshed under tight compute budgets. In such conditions, staged approaches accumulate mismatches [5]. Clusters become stale relative to embeddings; classifiers shift the embedding space but the index is not updated; retrieval improves for frequent classes while rare classes degrade because the representation collapses around dominant categories. These mismatches are not only accuracy issues. They affect system predictability, monitoring, and the ability to reason about failure modes.

A unified optimization perspective treats retrieval, clustering, and classification as coupled constraints on a shared representation. The shared representation is not merely a convenience but the central object [6] that determines system behavior: similarity neighborhoods drive retrieval and local density; local density shapes cluster formation; cluster structure and label structure jointly define a geometry that influences calibration, robustness, and fairness criteria. Once these tasks are treated jointly, new design questions appear. The objective weights determine whether clustering acts as a regularizer for retrieval or as a competing force that damages ranking. The choice of assignment model in clustering determines whether the embedding space be-

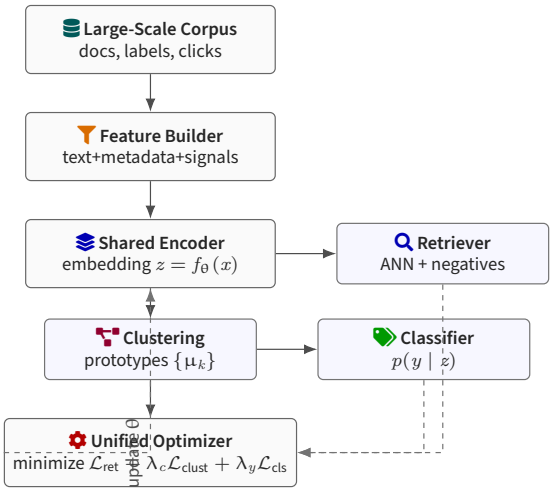


Figure 1. End-to-end unified framework: a shared encoder learns embeddings that simultaneously support approximate nearest-neighbor retrieval, prototype-based clustering, and supervised classification; the joint objective couples offline signals (labels) with interaction signals (clicks) for large-scale information systems.

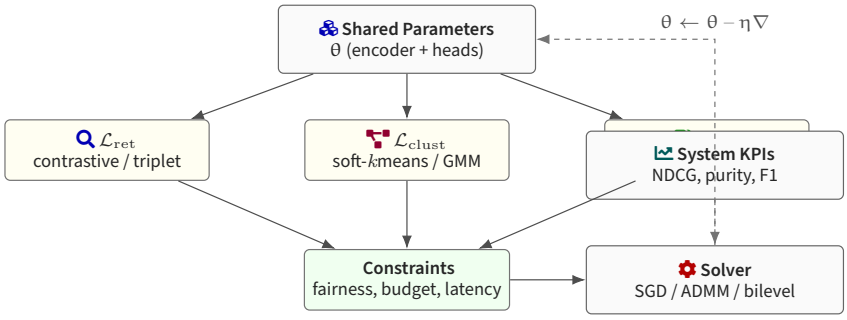


Figure 2. Unified optimization view: retrieval, clustering, and classification losses share parameters and are optimized under system-level constraints (e.g., latency and budget) while tracking multi-metric KPIs to balance relevance, structure discovery, and predictive accuracy.

comes piecewise linear around prototypes or remains smoothly structured [7]. The choice of classification head determines whether decision boundaries align with neighborhood connectivity or cut across it.

This paper develops a technical framework for joint learning that is grounded in optimization structure and scalable system constraints. The goal is not to assert that a single joint objective dominates all specialized solutions, but to provide a coherent set of modeling and algorithmic tools that make trade-offs explicit. The framework begins by formalizing a unified data model: a corpus of items, a stream of queries, partial labels, and latent cluster structure. Retrieval is modeled as a [8] ranking risk over query-item similarities. Clustering is modeled as a structured assignment of items to latent groups, with hard or soft constraints. Classification is modeled as a supervised risk, possibly with abstention or partial labeling. These components are coupled through a shared embedding function and shared constraints on representation scale and anisotropy.

The technical challenge is that naive joint optimization is unstable at scale. Retrieval losses are often contrastive and depend on negatives that must be sampled; clustering losses introduce assignment variables [9] whose updates can be expensive; classification losses are sensitive to label noise and imbalance. Moreover, system constraints require that the embedding function be deployable with bounded latency, that the index be refreshed at manageable intervals, and that training be distributed. Accordingly, the paper emphasizes optimization decompositions that support scalable computation, including block-coordinate updates, proximal steps, and constrained assignment solvers that can be batched. It also emphasizes the geometric effects of coupling, because geometry determines whether approximate nearest-neighbor search remains valid as the embedding evolves, and whether clusters remain stable enough to be useful [10].

The remainder of the paper proceeds as follows. A system setting and problem definition establish the joint learning variables and the operational constraints of large-scale retrieval systems. A unified objective is then derived, along with modeling options for ranking, clustering, and classification that share an embedding

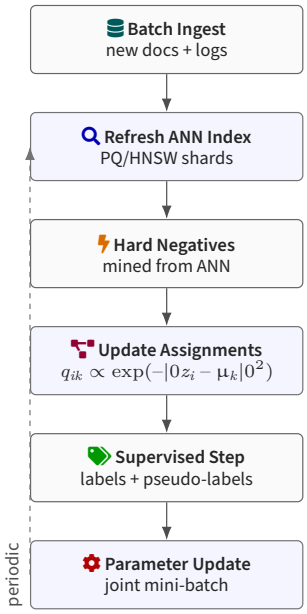


Figure 3. Alternating schedule for joint learning at scale: the ANN index is refreshed to mine informative hard negatives, clustering assignments are updated with current embeddings, and supervised steps incorporate labels and pseudo-labels before repeating the loop.

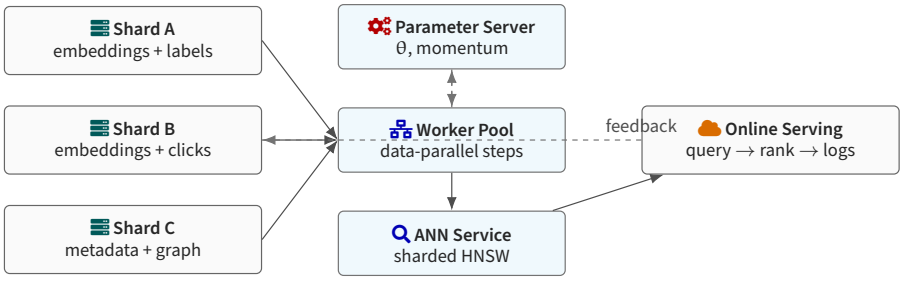


Figure 4. Large-scale deployment: sharded storage feeds a worker pool for joint optimization, synchronized by a parameter server; embeddings are exported to a sharded ANN service used by online serving, with user feedback routed back into training data streams.

space. Scalable algorithms are developed that combine stochastic gradient training with assignment updates and index-aware constraints. Theoretical considerations are discussed in terms of stability, geometry, and the interplay between [11] local neighborhoods and global partitions. Finally, practical deployment considerations are described, including streaming updates, delayed index refresh, and evaluation protocols that reflect the coupled nature of the tasks.

2. System Setting and Problem Definition

Consider a large corpus of items, each item represented by raw features such as text, images, metadata, or structured attributes. Items are retrieved in response to user queries, and queries may be of the same modality as items or may be multi-modal. Let the item set be denoted by [12] $\{x_i\}_{i=1}^{N}$, where N can range from millions to billions in industrial systems. Let the query distribution be represented by samples $\{q_j\}_{j=1}^{M}$, where M may also be large and may include repeated queries over time. Each query induces a relevance structure over items, which may be implicit, such as clicks, dwell time, or purchases, or explicit, such as curated relevance judgments.

In addition to retrieval interactions, the system may have labels for some items. Labels can represent categories, policy attributes, quality signals, or any supervised target relevant to downstream decisions. Label coverage is typically partial. Some items may have high-quality labels, [13] others may be weakly labeled, and many may be unlabeled. Let y_i denote a label for item i when available, and let $\mathcal{I}_\ell$ denote the index set of labeled items. The system also requires clustering, not necessarily into a fixed taxonomy, but often into a manageable number of groups for browsing, monitoring, or bootstrapping. Clustering can be fully unsupervised or weakly guided by labels, but it must operate over the full corpus and must be stable enough to support

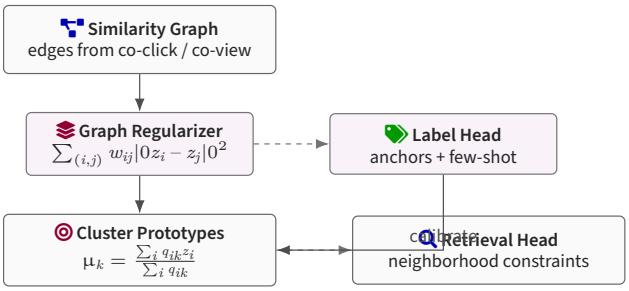


Figure 5. Structure-aware joint learning: a similarity graph induces a smoothness regularizer over embeddings, prototypes summarize clusters for scalable updates, and lightweight heads connect clustering structure to both classification calibration and retrieval neighborhood constraints.

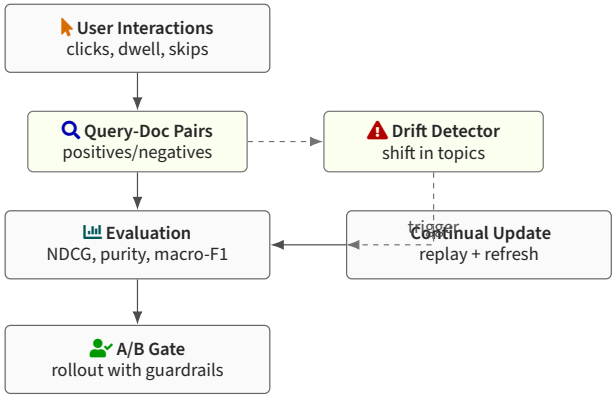


Figure 6. Closed-loop operation: interaction logs form training pairs, offline evaluation aggregates retrieval/clustering/classification metrics, drift detection flags distribution shifts, and continual updates are gated by controlled rollouts to stabilize large-scale system behavior.

longitudinal analysis.

The central modeling object is an embedding function that maps items and queries into a common space [14]. Let $f_\theta(\cdot)$ denote a parameterized encoder with parameters θ. For an item x_i, define its embedding as $z_i = f_\theta(x_i) \in \mathbb{R}^d$. For a query q_j, define $u_j = f_\theta(q_j) \in \mathbb{R}^d$ in the symmetric case, or use a separate query encoder g_ϕ when the modalities or requirements differ. Retrieval is then performed by similarity search, often based on inner product or cosine similarity. A typical choice is $s(u, z) = u^\top z$ with normalization of vectors to unit norm, which makes s proportional to cosine similarity. In large-scale systems, [15] retrieval is implemented via approximate nearest-neighbor indices, which assume that the embedding space is sufficiently stable and that distances preserve relevant neighborhoods.

Clustering introduces additional latent variables. A common model uses K cluster prototypes $p_k \in \mathbb{R}^d$ for $k \in \{1, \ldots, K\}$. Each item is assigned to one or more clusters. A hard assignment uses a mapping $c_i \in \{1, \ldots, K\}$. A soft assignment uses a vector $a_i \in \Delta^{K-1}$ where Δ^{K-1} is the probability simplex, so that $a_{ik} \geq 0$ and $\sum_{k=1}^{K} a_{ik} = 1$. In large corpora, hard assignments are compact and easy to store, while soft assignments provide smoother optimization but increase memory. A system can maintain [16] soft assignments during training and project to hard assignments for serving, or it can maintain hard assignments with periodic refinement.

Classification adds a head on top of embeddings. A linear head uses $W \in \mathbb{R}^{C \times d}$ for C classes, producing logits $\ell_i = Wz_i$ and probabilities via softmax. More complex heads are possible, but large-scale deployments often prefer simple heads for latency and maintainability. Classification interacts with retrieval because it shapes the embedding space. If classification encourages separability of labeled classes, the embedding space may become clustered [17] by label, which can help retrieval when labels correspond to user intent, but can harm retrieval when relevance depends on finer structure than the labels capture.

The joint learning problem is thus to learn θ (and possibly query encoder parameters), cluster prototypes $P = \{p_k\}$, assignment variables $A = \{a_i\}$, and classifier parameters W, in a way that serves three operational goals simultaneously. Retrieval performance must be strong under realistic query distributions and under approximate indexing. Clustering must be coherent and stable under corpus drift, while remaining computationally feasible at scale. Classification must achieve acceptable accuracy and calibration on labeled data,

Table 1. Notation used in the unified optimization framework.

Symbol	Description	Domain
$\mathcal{D}$	Training collection of items	$\{1, \dots, N\}$
$\mathcal{Q}$	Set of user queries	$\{1, \dots, Q\}$
Y	Set of class labels	$\{1, \dots, C\}$
$\mathbf{z}_i$	Joint latent representation of item i	$\mathbb{R}^k$
$\mathcal{L}$	Unified objective for all tasks	$\mathbb{R}$

Table 2. Summary of large-scale information system datasets used in experiments.

Dataset	#Items (10^6)	#Classes	#Queries (10^3)
WebSearch-L	12.4	1,000	520
ProductRank	4.7	5,000	310
NewsStream	2.1	800	150
ImageHub	1.8	2,000	220
LogClicks	9.3	120	600

while not degrading retrieval and clustering beyond acceptable bounds [18].

Operational constraints matter as much as objective definitions. The encoder must be deployable and efficiently evaluated. The embedding dimensionality d is limited by latency and memory, often in the range from tens to a few thousand. The index update schedule imposes a form of staleness: embeddings used by the index may lag behind current model parameters. This lag means that training must be robust to delayed feedback from retrieval performance, and that embedding updates should avoid abrupt geometric shifts that invalidate the index. Clustering must [19] be updated without full recomputation whenever possible. Classification must handle label drift, label noise, and the introduction of new classes without catastrophic disruption of the embedding geometry.

A unified optimization framework should expose these constraints as explicit terms or implicit design choices, rather than leaving them as post-hoc engineering. The next section defines a joint objective that couples retrieval, clustering, and classification through shared embeddings, while admitting algorithmic decompositions aligned with system constraints.

3. Unified Objective and Optimization Geometry

A joint formulation begins with a shared embedding model and defines task [20] losses that act on the same embeddings. The general form is a composite objective with constraints that capture normalization, assignment structure, and regularization. Let $\mathcal{D}_{\text{ret}}$ denote a set of retrieval interactions, each interaction consisting of a query q, a set of relevant items $\mathcal{P}(q)$, and a set of sampled non-relevant items $\mathcal{N}(q)$. Let $\mathcal{D}_{\text{cls}}$ denote labeled items. Let $\mathcal{D}_{\text{all}}$ denote all items. The optimization variables include encoder parameters θ, cluster prototypes P, assignment variables A, and classifier parameters W.

A unified objective can be written as a weighted sum, augmented with constraints:

$$
\begin{aligned}
\min_{\theta,\, W,\, P,\, A} \quad & \lambda_r \mathcal{L}_{\text{ret}}(\theta)\,[21] \\
& + \lambda_c \mathcal{L}_{\text{clu}}(\theta,\ P,\ A) \\
& + \lambda_s \mathcal{L}_{\text{cls}}(\theta,\ W) \\
& + \lambda_u \mathcal{L}_{\text{unsup}}(\theta) \\
& + \lambda_g \mathcal{R}(\theta,\ W,\ P,\ A)\,[22] \\
\text{s.t.} \quad & A\mathbf{1} = \mathbf{1} \\
& A \geq 0 \\
& |0p_k|0_2 = 1 \ \ \forall k \\
& |0z_i|0_2 = 1 \ \ \forall i \in \mathcal{I}_{\text{norm}}
\end{aligned}
$$

Here $\lambda_r, \lambda_c, \lambda_s, \lambda_u, \lambda_g$ are nonnegative weights [23]. The constraint set includes simplex constraints on assignments, and optional unit-norm constraints on prototypes and embeddings. The set $\mathcal{I}_{\text{norm}}$ can include all items, or only those used for retrieval, depending on whether the system requires cosine similarity.

The similarity s can be inner product or cosine similarity. The temperature τ controls the sharpness of the distribution and therefore the effective margin [24]. The negative set can include in-batch negatives, mined hard negatives, or a mixture. The retrieval loss shapes local neighborhoods by pulling positives close to queries while pushing negatives away. Its geometry depends on normalization. With unit norms and inner product, the loss acts on angles, encouraging separation by angular margins rather than Euclidean distances.

Clustering can be expressed through prototypes and assignments. A soft K-means-like objective uses squared

Table 3. Architectural variants of the unified framework evaluated in this work.

Variant	Shared Encoder	Optimization	Regularization
Decoupled	No	Task-specific SGD	ℓ_2 only
Shared-Linear	Yes	Alternating SGD	ℓ_2 + orthogonality
Shared-Nonlinear	Yes	Joint SGD	dropout + ℓ_2
Multi-Head	Yes	Joint SGD	entropy + contrastive

Table 4. Asymptotic complexity comparison of baseline and unified methods.

Method	Training Complexity	Inference Complexity
BM25	$\mathcal{O}(N)$	$\mathcal{O}(\log N)$
k-Means + SVM	$\mathcal{O}(Nkd)$	$\mathcal{O}(kd)$
Triplet Network	$\mathcal{O}(Nbd)$	$\mathcal{O}(d)$
Unified (ours)	$\mathcal{O}(Nd + Nk)$	$\mathcal{O}(k)$

distances and an entropy term that [25] controls assignment softness:

$$\mathcal{L}_{\text{clu}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} a_{ik}\,|0z_i - p_k|0_2^2 + \gamma\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} a_{ik}\log a_{ik}$$

The entropy term with coefficient $\gamma > 0$ encourages spread assignments when γ is large, and approaches hard assignments as γ decreases. When embeddings and prototypes are normalized, the squared distance can be replaced by a cosine dissimilarity. The assignment variables can be optimized in closed form given prototypes and [26] embeddings, producing a softmax over negative distances. However, closed form updates can lead to degenerate cluster sizes. To control cluster balance, a global constraint on cluster mass can be added, which converts assignment updates into a constrained optimization problem.

Classification can be represented as a cross-entropy loss over labeled items. For labeled item i with label $y_i \in \{1, \ldots, C\}$:

$$\mathcal{L}_{\text{cls}} = \frac{1}{|\mathcal{I}_\ell|}\sum_{i\in\mathcal{I}_\ell} -\log\frac{\exp\left(w_{y_i}^\top z_i\right)}{\sum_{c=1}^{C}\exp\left(w_c^\top z_i[27]\right)}$$

When labels are noisy or partial, the loss can be modified to use label smoothing, partial label sets, or confidence weights. The classifier head can also be tied to cluster prototypes when clusters are intended to approximate classes, but tying is optional and can be overly restrictive when clusters represent latent semantics beyond labels.

An unsupervised representation loss can be included to stabilize training on unlabeled data, for example by using self-distillation between augmentations. The role of $\mathcal{L}_{\text{unsup}}$ in a joint system is often not to improve a single metric, but to reduce [28] overfitting of the embedding to whichever supervised signal is most available. When retrieval interactions are sparse relative to the corpus, or when labels cover only a small fraction, unsupervised structure can prevent representation collapse and can maintain smoothness in regions of the space not directly constrained by retrieval or classification.

The regularizer $\mathcal{R}$ captures system-level constraints. A particularly relevant family of regularizers controls anisotropy and feature norm growth. Embedding anisotropy can harm approximate nearest neighbor search because most points concentrate in a narrow cone, making distance distributions less informative. An anisotropy penalty can be [29] based on the covariance of embeddings in a batch, encouraging a spread spectrum rather than a dominant principal component. Another regularizer controls prototype separation, preventing cluster collapse into a few prototypes that dominate assignment. Regularizers can also encourage alignment between class weights and cluster prototypes if interpretability or stability is desired.

Joint optimization introduces geometric coupling. Retrieval wants local neighborhoods aligned with relevance judgments, often requiring fine granularity [30]. Clustering wants global partitions that reduce intracluster variance and separate cluster centroids. Classification wants decision boundaries that separate labels, which may or may not correspond to cluster structure. When these objectives are simultaneously active, the embedding geometry can shift toward a compromise that depends on weights and on the relative scale of gradients from each loss.

To understand coupling, it is useful to examine gradient contributions. Let z_i be an item embedding. The retrieval loss contributes gradients for items that appear as positives or negatives relative to queries [31]. The clustering loss contributes a gradient that points from z_i toward a weighted sum of prototypes, propor-

Table 5. Retrieval performance on WebSearch-L averaged over all queries.

Method	MAP@10	NDCG@10	Recall@100
BM25	0.412	0.463	0.612
Dual Encoder	0.528	0.587	0.721
ColBERT	0.561	0.612	0.745
Decoupled Multi-Task	0.547	0.599	0.736
Unified (ours)	**0.593**	**0.644**	**0.781**

Table 6. Clustering quality on latent representations learned under different methods.

Method	NMI	ARI	Clustering F1
k-Means (features)	0.392	0.271	0.433
Spectral Clustering	0.417	0.298	0.451
DeepCluster	0.468	0.332	0.497
Unified (ours)	**0.521**	**0.381**	**0.542**

tional to its assignment. The classification loss contributes a gradient that points toward the weight vector of the true class and away from others. If these directions align, joint optimization is stable. If they conflict, oscillations can occur, and the embedding can become distorted, for example by stretching along label directions while collapsing along semantic directions needed for retrieval.

A practical unified framework therefore needs mechanisms to manage gradient conflict and to control the relative influence of each task as a [32] function of data availability and system constraints. One approach is to adapt weights dynamically, for example by normalizing gradient norms across tasks. Another approach is to constrain representations, for example by unit-norm embeddings and bounded temperature, which prevents the model from satisfying a loss by unbounded scaling. Yet another approach is to introduce auxiliary variables that decouple tasks, such as maintaining separate but coupled embeddings for retrieval and classification, linked by a penalty that keeps them close. However, [33] separate embeddings increase serving complexity and index maintenance cost. The framework in this paper emphasizes shared embeddings but uses constraints and block updates to keep optimization stable.

A constrained assignment model can further improve stability. Suppose the system wants cluster sizes to remain within a range, because extremely small clusters are unstable and extremely large clusters reduce usefulness. A global mass constraint can enforce approximate balance:

$$\min_{A} \quad \sum_{i=1}^{N}\sum_{k=1}^{K} a_{ik}\, d_{ik} + \gamma \sum_{i=1}^{N}\sum_{k=1}^{K} a_{ik} \log a_{ik}$$

[34]s.t. $$\sum_{k=1}^{K} a_{ik} = 1 \;\; \forall i$$

$$\sum_{i=1}^{N} a_{ik} = m_k \;\; \forall k$$

$$a_{ik} \geq 0 \;\; \forall i, k$$

Here d_{ik} is a dissimilarity between z_i and p_k, and m_k is a target mass for cluster k. This is an entropy-regularized optimal [35] transport problem that can be solved by iterative matrix scaling. Such a constraint prevents collapse and yields more predictable clusters, though it introduces computation. In practice, the assignment can be computed on minibatches with approximate mass constraints, or with moving estimates of cluster mass.

The unified objective thus becomes a structured composite optimization problem with smooth terms (retrieval and classification), a structured term (clustering with constraints), and system regularizers. The next section develops algorithms that exploit this structure for scalability [36] and stability in large-scale systems.

4. Algorithms for Large-Scale Deployment

A key requirement is that training be feasible when N is extremely large, retrieval interactions are numerous, and clustering assignments cannot be recomputed from scratch at every step. The unified problem suggests block-coordinate approaches, where different variable blocks are updated with methods suited to their structure. The encoder parameters θ and classifier weights

Table 7. Classification performance on the ProductRank dataset.

Method	Top-1 Acc	Top-5 Acc	Macro-F1
Logistic Regression	78.3%	93.5%	74.1%
Deep Classifier	82.7%	95.8%	79.4%
Decoupled Multi-Task	83.2%	96.1%	80.2%
Unified (ours)	**85.6%**	**97.3%**	**82.8%**

Table 8. Ablation study of the unified objective on WebSearch-L.

Loss Configuration	MAP@10	NMI	Top-1 Acc
$\mathcal{L}_{\text{retr}}$ only	0.574	0.403	81.9%
$\mathcal{L}_{\text{retr}} + \mathcal{L}_{\text{cls}}$	0.583	0.416	84.1%
$\mathcal{L}_{\text{retr}} + \mathcal{L}_{\text{clu}}$	0.579	0.447	82.4%
Full $\mathcal{L}_{\text{joint}}$	**0.593**	**0.521**	**85.6%**

W are typically updated by stochastic gradient descent or variants. Prototypes P can be updated by moving averages or gradient steps [37]. Assignments A can be updated by closed-form soft assignments or by constrained solvers. The challenge is to combine these updates without introducing instability due to delayed or noisy assignment signals.

A practical training loop can be structured as alternating phases. During an encoder update phase, minibatches of queries and items are sampled. Retrieval loss is computed using in-batch negatives and any available mined negatives. Classification loss is computed for labeled items in the batch [38]. Clustering loss is computed using current prototypes and either cached assignments or on-the-fly assignments for the batch. Gradients are backpropagated through f_θ and through the classification head. Prototypes can be updated via gradients or via separate accumulation. Assignments can be updated less frequently, using either batch-wise computations or periodic full passes.

When clustering is implemented with soft assignments without global mass constraints, assignments for a minibatch can be computed as:

$$a_{ik} = \frac{\exp\big([39] - d(z_i,\ p_k)/\gamma\big)}{\sum_{t=1}^{K} \exp\big(-d(z_i,\ p_t)/\gamma\big)}$$

This update is cheap, differentiable, and amenable to GPU computation. However, it can produce imbalanced clusters when prototypes drift toward high-density regions. To mitigate this, prototypes can be updated with constraints that promote separation, or the model can include a penalty on cluster mass deviation estimated in minibatches [40].

When global mass constraints are important, minibatch optimal transport can be used. Let a minibatch contain B items. Define a cost matrix $D \in \mathbb{R}^{B \times K}$ with entries $D_{ik} = d(z_i, p_k)$. Let row marginals be uniform and column marginals correspond to desired cluster proportions. An entropy-regularized transport plan can be found by iterative scaling. The computation cost scales with BK per iteration. For large K, this is still feasible when B is moderate, and when the number of scaling iterations is limited [41]. The resulting assignments are more balanced and can improve cluster stability, which in turn reduces oscillations in prototype updates.

Prototype updates can be performed via a moving average consistent with the assignment model. For soft assignments, a prototype can be updated as a normalized weighted average of embeddings:

$$p_k \leftarrow \mathrm{Norm}\Big((1 - \eta_p)\, p_k + \eta_p \frac{[42] \sum_{i \in \mathcal{B}} a_{ik}\, z_i}{\sum_{i \in \mathcal{B}} a_{ik} + \epsilon}\Big)$$

Here η_p is a prototype step size, $\mathcal{B}$ is the current minibatch, ϵ prevents division by zero, and Norm projects to the unit sphere when cosine similarity is used. This update decouples prototype learning from encoder gradients and can be more stable than jointly optimizing prototypes with gradients, especially when clustering weights are large.

The encoder update integrates retrieval, clustering, and classification losses. Let $\mathcal{L}(\theta)$ denote the batch loss. A gradient step with learning rate η is: [43]

$$\theta \leftarrow \theta - \eta\, \nabla_\theta \Big(\lambda_r\, \mathcal{L}_{\text{ret}} + \lambda_c\, \mathcal{L}_{\text{clu}} + \lambda_s\, \mathcal{L}_{\text{cls}} + \lambda_u\, \mathcal{L}_{\text{unsup}}\Big)$$

In practice, the gradients from each term can have different scales and [44] different variance. Retrieval gradients can be noisy due to negative sampling. Clustering gradients can be dense because every item receives a prototype pull. Classification gradients can be sparse if labels are rare in the batch. To manage these differences, gradient scaling strategies are often necessary. One approach is to normalize each task gradient by its running norm estimate. Another approach is to tune weights to target [45] comparable contributions to embedding updates. The goal is not to equalize loss magnitudes, which can be misleading, but to control

Table 9. Training hyperparameters for the unified optimization framework.

Hyperparameter	Symbol	Value	Notes
Latent dimension	k	256	Shared across tasks
Batch size	b	512	Uniform for all datasets
Retrieval weight	λ_{retr}	1.0	Base task
Clustering weight	λ_{clu}	0.5	Encourages compactness
Classification weight	λ_{cls}	0.7	Balances decision boundary
Learning rate	η	3×10^{-4}	Adam optimizer
Temperature	τ	0.07	For contrastive terms

embedding drift in directions that matter for index stability and neighborhood preservation.

Index stability introduces additional considerations. In a deployed system, the retrieval index is built from embeddings computed by a particular model snapshot. During training, the embedding function changes. If changes are too abrupt, the index becomes stale and offline [46] retrieval evaluation no longer matches online performance. A unified framework can incorporate a regularizer that penalizes deviation from a reference embedding function over a subset of anchor items. Let θ_0 denote a reference snapshot used to build the current index. Define anchor items $\mathcal{A}$ sampled from the corpus. An index stability regularizer is:

$$\mathcal{R}_{\text{stab}} = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} |0 f_\theta(x_i) - f_{\theta_0}(x_i)|0_2^2 [47]$$

This term encourages gradual embedding drift, enabling less frequent index refresh. It also implicitly couples training to operational constraints. The reference snapshot θ_0 can be updated on a schedule aligned with index rebuilds. When θ_0 is updated, the penalty resets, allowing the model to explore new geometry while keeping drift controlled between refreshes.

Large-scale retrieval training often uses hard negative mining, which is expensive when the embedding space changes. Joint training with clustering [48] can assist mining by providing prototype neighborhoods that approximate semantic regions. For a query, candidate negatives can be sampled from clusters adjacent to the positive cluster, increasing hardness without global search. Conversely, retrieval signals can refine clusters by providing local neighborhood constraints derived from user interactions. This mutual reinforcement is a core motivation for joint training, but it requires care to avoid feedback loops where early cluster errors bias negative sampling and reinforce incorrect structure. A practical approach is to delay strong cluster-guided mining until prototypes stabilize, or [49] to mix uniform negatives with cluster-guided negatives at a controlled ratio, such as 30% cluster-guided and 70% uniform early in training, shifting toward higher cluster-guided ratios as stability improves.

Distributed training introduces further constraints. Encoders are trained with data parallelism. Prototypes and assignment statistics can be stored and updated via parameter servers or replicated with periodic synchronization. If prototypes are updated frequently, stale prototypes across workers can create inconsistency in clustering gradients. A remedy [50] is to update prototypes using synchronized reductions at a fixed cadence, or to use exponential moving averages that are less sensitive to staleness. Assignments can be computed locally per minibatch, but global mass constraints require global statistics. Approximate solutions can be built from moving estimates of cluster mass, updated with distributed aggregations, avoiding full transport solves across the full dataset.

The computational footprint of joint training must be compared to the footprint of separate pipelines. Joint training adds clustering compute and possibly additional unsupervised terms. However, it can reduce the [51] need for repeated offline clustering and repeated retraining of separate heads. The net effect depends on how often models are refreshed, how frequently the corpus changes, and how expensive it is to rebuild indices. In many systems, index rebuild cost dominates. If joint training with stability regularization reduces index rebuild frequency from daily to every two or three days, the saved compute can offset the added cost of clustering updates, even when clustering uses constrained assignment solvers.

A final algorithmic aspect is handling new classes and new clusters over time. Classification may require new labels; clustering may require new [52] prototypes to capture emerging topics. A unified framework can treat prototype creation as a split operation when a cluster becomes too heterogeneous, and prototype removal as a merge operation when two clusters converge. These operations can be triggered by statistics such as within-cluster variance or assignment entropy. While the details are system dependent, the optimization view clarifies that splits and merges are modifications of the feasible set, requiring careful reinitialization to avoid disrupting retrieval neighborhoods.

The next section discusses theoretical considerations that guide weight [53] selection, stability design, and the interpretation of joint objectives in terms of em-

bedding geometry and neighborhood preservation.

5. Theoretical Considerations: Stability, Trade-offs, and Geometry

Joint objectives are often criticized for creating trade-offs that are difficult to predict. A unified optimization framework does not eliminate trade-offs, but it makes them analyzable. The core object of analysis is the embedding geometry induced by the combined loss. Geometry governs retrieval because nearest-neighbor structure is geometric. It governs clustering because assignments depend on distances to prototypes [54]. It governs classification because decision boundaries partition the space. Theoretical considerations therefore focus on how the combined gradients shape local neighborhoods and global structure.

A useful concept is neighborhood stability under updates. Consider an item embedding z_i and its set of nearest neighbors under the current model. Retrieval quality depends on whether relevant items remain among the nearest neighbors of queries. Approximate indexing assumes that small embedding perturbations do not reorder neighbors too much. A first-order approximation can relate perturbations to changes in similarity margins [55]. Let z_i and z_j be two candidate items for a query embedding u. The margin is $\Delta = s(u, z_i) - s(u, z_j)$. Under perturbations $\delta u, \delta z_i, \delta z_j$, the margin changes by approximately:

$$\delta\Delta = (\delta u)^\top (z_i - z_j) + u^\top (\delta z_i - \delta z_j)$$

If the original margin is small, small perturbations can flip the order. Joint training increases the sources of perturbation because clustering and classification both update z_i even when i is not directly involved in a retrieval interaction [56]. Stability regularizers and conservative learning rates can keep perturbations small, but this may slow progress. A more nuanced approach is to design clustering and classification losses that respect retrieval margins, for example by using angular constraints compatible with cosine similarity, or by restricting prototype pulls to be orthogonal to directions that matter for a set of anchor queries.

Trade-offs can be formalized via multi-objective optimization. The weighted sum objective approximates a Pareto frontier, but weights do not directly correspond to desired metric trade-offs because metrics are nonlinear and depend on indexing approximations. Nonetheless, weights control the relative gradient contributions [57]. A practical theoretical view is to analyze the alignment between task gradients. Let $g_r = \nabla_\theta \mathcal{L}_{\text{ret}}$, $g_c = \nabla_\theta \mathcal{L}_{\text{clu}}$, and $g_s = \nabla_\theta \mathcal{L}_{\text{cls}}$. If dot products $\langle g_r, g_c \rangle$ and $\langle g_r, g_s \rangle$ are positive on average, tasks are synergistic. If they are negative, tasks conflict. While exact computation is expensive, minibatch estimates can provide a signal. A system can then adapt weights to reduce conflict, for example by decreasing λ_c when $\langle g_r, g_c \rangle$ is consistently negative, or by modifying clustering to act on a slower timescale so that it shapes global structure without interfering with [58] retrieval fine-tuning.

Clustering introduces the risk of prototype collapse and representation degeneracy. If prototypes become too close, clusters lose meaning and assignments become unstable. If embeddings collapse around a few directions, retrieval becomes less discriminative. Collapse can occur when the clustering loss is too strong relative to retrieval and classification, or when the assignment entropy is too low early in training [59]. A theoretical mitigation is to ensure that the clustering objective is strongly convex in assignments for fixed embeddings and prototypes, which is achieved by sufficient entropy regularization. Another mitigation is to impose a minimum separation between prototypes. For unit-norm prototypes, a separation penalty can be:

$$\mathcal{R}_{\text{sep}} = \sum_{k=1}^{K} \sum_{t \neq k} \max\bigl(0,\ p_k^\top p_t - \alpha\bigr)^2$$

Here α is a threshold controlling how close prototypes are allowed to be. This term encourages prototypes to spread on the sphere, reducing [60] collapse risk. It also supports approximate indexing by maintaining a more uniform embedding distribution, which can improve the quality of vector quantization schemes used by indices.

Classification introduces its own geometric biases. A linear classifier on unit-norm embeddings effectively encourages embeddings to align with class weight directions. If the number of classes is small relative to the embedding dimension, embeddings may concentrate around a few weight vectors, reducing within-class diversity. This is acceptable when labels [61] capture the semantic granularity needed for retrieval, but harmful when retrieval requires fine structure within a class. One theoretical response is to decouple classification from the embedding by using a temperature or margin that avoids overly sharp alignment. Another is to use a hierarchical classification head where coarse labels influence global geometry while a weaker loss preserves local structure. In the unified framework, such choices correspond to modifying $\mathcal{L}_{\text{cls}}$ and its coupling weight.

The interaction between clustering and classification depends on whether clusters are intended to align with classes [62]. If classes are a subset of semantic structure, clusters can represent subtopics within classes, and a mild alignment constraint can improve interpretability. If classes are orthogonal attributes, forcing alignment can be counterproductive. A flexible approach is to allow soft coupling between class weights and prototypes via a penalty that encourages

proximity without identity:

$$\mathcal{R}_{\text{align}} = \sum_{c=1}^{C} \min_{k \in \{1,\dots,K\}} |0 w_c - \beta p_k|0_2^2$$

Here β adjusts scale. This penalty encourages each class direction to be near some prototype, but does not force a one-to-one mapping [63]. In practice, the minimum operation can be approximated by a softmin to keep the objective smooth. The penalty can be applied selectively to classes with sufficient label support to avoid over-constraining rare classes.

Another theoretical issue is identifiability and invariance. Embedding spaces have rotational symmetries when objectives depend only on pairwise similarities. Clustering breaks some of these symmetries by anchoring prototypes, and classification breaks symmetries by anchoring class weights. This anchoring can be beneficial because it reduces [64] arbitrary drift, which helps index stability. However, it can also create undesired dependence on initialization. A stability perspective suggests that anchoring should be introduced gradually. Early in training, allow embeddings to form neighborhoods driven by retrieval and unsupervised structure. Later, strengthen clustering and classification coupling to impose global structure. This corresponds to a continuation method where λ_c and λ_s vary over time [65]. The theoretical justification is that nonconvex objectives can have better optimization trajectories when constraints are introduced after a stable manifold has formed.

Finally, large-scale systems face distribution drift. Drift changes the objective landscape over time. A joint framework can address drift by including time-dependent regularization, for example by penalizing rapid changes in prototypes and class weights, or by using online convex optimization ideas for the last-layer parameters while updating the encoder more slowly. The resulting two-timescale dynamics can stabilize [66] serving: the index can be refreshed on a slower schedule while last-layer adjustments handle recent data. Although exact guarantees are limited in nonconvex settings, the optimization decomposition clarifies which parts of the model are allowed to move quickly without breaking retrieval neighborhoods.

The next section turns from theory to system practice, focusing on how joint training interacts with indexing, streaming data, monitoring, and evaluation in large-scale deployments.

6. Practical Considerations and Empirical Protocols

A unified optimization framework is only useful if it maps to deployable mechanisms. Practical considerations begin with data [67]. Retrieval interactions are often biased toward popular items and frequent queries. Labels are sparse and may be biased toward curated subsets. Clustering must cover the full corpus, including long-tail items with few or no interactions. A joint system therefore requires careful sampling to avoid training the embedding only on the head of the distribution. One approach is to include a fixed fraction of uniformly sampled items in each batch for clustering and unsupervised losses, while retrieval pairs and labeled items occupy the remaining fraction. The exact ratios are [68] system dependent, but the guiding principle is to ensure that the embedding geometry is shaped by both interaction-rich and interaction-poor regions of the corpus.

Indexing constraints shape training schedules. Approximate nearest-neighbor indices are built on a snapshot of embeddings. If embeddings drift, the index becomes suboptimal. Some systems rebuild indices frequently; others rely on incremental updates. Joint training can help reduce rebuild frequency by incorporating stability regularizers, but this introduces a [69] trade-off between immediate improvement and operational consistency. A practical protocol is to align training epochs with index refresh windows. During a window, the model is trained with a stability penalty relative to the deployed snapshot, allowing incremental improvement without large drift. At refresh time, the index is rebuilt with the new snapshot, and the stability reference is updated. This protocol makes the relationship between optimization and operations explicit.

Clustering deployment raises questions about where clusters are used. Clusters can drive navigation and discovery, serve as a compression of [70] semantic space for monitoring, or act as a prior for retrieval. Different uses require different cluster properties. For navigation, interpretability and stability may matter more than within-cluster variance. For retrieval priors, alignment with relevance neighborhoods matters. The unified framework supports these variants by adjusting clustering objectives and constraints. For example, interpretability can be encouraged by prototypes [71] that correspond to real items, achieved by restricting prototypes to be chosen from embeddings of items. Stability can be encouraged by temporal regularization on assignments, penalizing frequent reassignment. Alignment with retrieval can be encouraged by adding a term that pulls prototypes toward regions frequently used in query neighborhoods.

Monitoring and debugging are essential in large-scale systems. Joint training complicates attribution: when retrieval degrades, it may be due to clustering weight, classification label drift, or changes in negative sampling. A practical monitoring approach is to track not only task [72] metrics but also geometric diagnostics. Diagnostics include embedding norm statistics, anisotropy measures such as the ratio of leading singular values, cluster mass distributions, prototype separation statistics, and neighborhood overlap between

consecutive snapshots. These diagnostics help detect collapse, drift, and instability earlier than end metrics alone. They also provide actionable levers, because each diagnostic corresponds to a term or constraint in the unified objective.

Evaluation protocols must reflect coupling. Retrieval evaluation usually measures ranking metrics on a test set of queries [73]. Classification evaluation measures accuracy or calibration on labeled items. Clustering evaluation measures internal criteria or agreement with weak labels. In a joint system, these evaluations should be performed not only in isolation but also under realistic serving conditions. Retrieval should be evaluated with the actual approximate index configuration, because small geometric shifts can have outsized effects on approximate search. Classification should be evaluated on items retrieved under the index, because distribution shift between full-corpus and retrieved subsets can affect calibration [74]. Clustering should be evaluated longitudinally, because cluster stability over time is often a primary operational requirement.

A recurring empirical issue is weight selection. Because the unified loss combines heterogeneous terms, naive tuning can be inefficient. A practical approach is to begin with a retrieval-strong baseline, then introduce clustering and classification gradually. The introduction can be scheduled such that the embedding space first forms meaningful local neighborhoods, after which clustering shapes global organization and classification imposes label-consistent directions. This schedule reduces [75] early conflict and can improve stability. Another approach is to use adaptive weighting based on gradient norms or on validation metrics. Adaptive approaches must be designed carefully to avoid oscillations, especially when validation metrics are noisy due to approximate indexing.

Another practical issue is handling missing labels and weak supervision. In many systems, labels can be inferred from user behavior but are noisy. A joint framework can incorporate weak labels as soft constraints in classification, while clustering provides a regularizing structure that prevents the model from overfitting to noisy labels. Conversely, classification can guide clustering by anchoring certain regions of the [76] space. However, over-anchoring can reduce discovery of latent structure. A balanced approach is to treat labels as one of several signals, with clustering and unsupervised losses preserving flexibility. The weight on classification can be increased when label confidence is high and reduced when confidence is low, which can be implemented by per-example weights in the classification loss.

Scalability also depends on the choice of K and on memory for assignments. Storing full soft assignments for billions of items is impractical. A common strategy is to store only hard assignments and [77] prototype statistics, and to compute soft assignments on the fly for minibatches during training. Another strategy is to store sparse soft assignments, keeping only the top few cluster probabilities per item. In the optimization view, sparse assignments correspond to restricting A to a sparse subset of the simplex, which can be approximated by temperature settings or by explicit top-m truncation. Such truncation must be handled carefully because it introduces non-smoothness, but in practice it can be stable when prototypes change slowly.

Finally, the unified framework must be compatible with privacy and policy constraints that often govern large-scale information systems [78]. Even when explicit constraints are not modeled mathematically, the joint objective can incorporate penalties that reduce memorization of rare interactions, or that prevent overly sharp decision boundaries that may amplify noise. These considerations are system specific, but the optimization framework provides a place to encode them as regularizers or as constraints on updates.

The conclusion synthesizes the unified view and summarizes how it supports practical design choices without assuming that a single joint solution dominates specialized pipelines.

7. Conclusion

This paper presented a unified optimization framework for joint retrieval, clustering, and classification in large-scale information systems [79]. The framework treats these tasks as coupled objectives acting on shared embeddings, shared prototypes, and structured assignment variables, augmented by constraints that reflect operational requirements such as normalization, cluster balance, and index stability. Retrieval was modeled as a ranking-oriented contrastive risk, clustering as prototype-based assignment with optional global constraints, and classification as a supervised risk over labeled subsets. The resulting objective was shown to admit scalable decompositions: stochastic gradient updates for encoder and classifier parameters, moving-average or gradient updates for prototypes, and closed-form or constrained solver updates for assignment [80] variables.

A central theme was the geometric coupling induced by joint learning. Retrieval depends on neighborhood structure, clustering imposes global partitions, and classification shapes decision-aligned directions in the embedding space. The framework makes these influences explicit and suggests mechanisms for stability, including entropy-regularized assignments, prototype separation penalties, and index-aware drift regularizers. Algorithmic discussion emphasized large-scale constraints, including approximate nearest-neighbor indexing, distributed training, and streaming updates, and it described practical protocols that align [81] training schedules with index refresh windows.

The unified view does not remove the need for empirical tuning, and it does not imply that joint training is always preferable to staged pipelines. Instead, it provides a structured language for reasoning about tradeoffs, selecting coupling strengths, and designing optimization procedures that respect system constraints. In settings where retrieval, clustering, and classification must coexist over the same evolving corpus, the framework offers a principled basis for constructing joint learners whose behavior can be monitored, stabilized, and adapted as operational requirements change [82].

References

[1] R. Malik, S. Kim, X. Jin, C. Ramachandran, J. Han, I. Gupta, and K. Nahrstedt, "Mlr-index: An index structure for fast and scalable similarity search in high dimensions," in International Conference on Scientific and Statistical Database Management, pp. 167–184, Springer, 2009.

[2] M. Wang, F. He, H. Liu, X. Lin, H. Yu, and C. Li, "Multi-objective optimization of distributed energy systems under uncertainty," in 2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2), pp. 1670–1674, IEEE, 10 2020.

[3] Y. You, Y. He, S. Rajbhandari, W. Wang, C.-J. Hsieh, K. Keutzer, and J. Demmel, "Fast lstm by dynamic decomposition on cloud and distributed systems," Knowledge and Information Systems, vol. 62, pp. 4169–4197, 7 2020.

[4] S. Dokos, Distributed Systems Modelling with Partial Differential Equations, pp. 105–157. Springer Berlin Heidelberg, 3 2017.

[5] H. L. Alexander, "State estimation for distributed systems with sensing delay," in SPIE Proceedings, vol. 1470, pp. 103–111, SPIE, 8 1991.

[6] S. B. Ahsan and I. Gupta, "Dcc@podc - the cat theorem and performance of transactional distributed systems," in Proceedings of the 4th Workshop on Distributed Cloud Computing, pp. 6–6, ACM, 7 2016.

[7] R. Achar, P. Dawn, and C. V. Lopes, "Onward! - gotcha: an interactive debugger for got-based distributed systems," in Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, pp. 94–110, ACM, 10 2019.

[8] S. Belforte, A. Fanfani, I. Fisk, J. Flix, J. M. Hernandez, T. Kress, J. Letts, N. Magini, V. Miccio, and A. Sciabà, "Bringing the cms distributed computing system into scalable operations," Journal of Physics: Conference Series, vol. 219, pp. 062015–, 4 2010.

[9] A. W. Rodriguez and J. A. Reggia, "A distributed learning algorithm for particle systems," Integrated Computer-Aided Engineering, vol. 16, pp. 1–20, 1 2009.

[10] M. Perry, "Socially distributed cognition in loosely coupled systems," AI & SOCIETY, vol. 25, pp. 387–400, 1 2010.

[11] S. Li and Y. Zheng, Distributed Model Predictive Control for Plant-Wide Systems - Cooperative Distributed Predictive Control. Wiley, 8 2015.

[12] J.-S. Duan and Y. Li, "Identification of system with distributed-order derivatives," Fractional Calculus and Applied Analysis, vol. 24, pp. 1619–1628, 10 2021.

[13] T. Srinivasan, R. Chandrasekar, V. Vijaykumar, V. Mahadevan, A. Meyyappan, and A. Manikandan, "Localized tree change multicast protocol for mobile ad hoc networks," in 2006 International Conference on Wireless and Mobile Communications (ICWMC'06), pp. 44–44, IEEE, 2006.

[14] L. G. Shattuck, "Communication of intent in distributed supervisory control systems," Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 41, pp. 259–263, 10 1997.

[15] M. Fischer, W. Lamersdorf, J. Liebeherr, and M. Mühlhäuser, "Introduction to the special section on recent advances in networks and distributed systems," ACM Transactions on Internet Technology, vol. 22, pp. 1–3, 11 2022.

[16] Z. Lan, V. Taylor, and G. L. Bryan, "Dynamic load balancing of samr applications on distributed systems," Scientific Programming, vol. 10, pp. 319–328, 11 2002.

[17] Y. Song, X. He, Z. Liu, W. He, C. Sun, and F.-Y. Wang, "Parallel control of distributed parameter systems," IEEE transactions on cybernetics, vol. 48, pp. 3291–3301, 7 2018.

[18] K. Hines and G. Borriello, "A geographically distributed framework for embedded system design and validation," in Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175), IEEE, 11 2002.

[19] C. H. Lin and W. Wolf, "A case study in clock synchronization for distributed camera systems," SPIE Proceedings, vol. 5683, pp. 164–174, 3 2005.

[20] K. L. Stamber, A. Kelic, R. Taylor, J. M. Henry, and J. E. Stamp, "Distributed energy systems: Security implications of the grid of the future," 1 2017.

[21] K. R. Jayaram and P. Eugster, "Debs - program analysis for event-based distributed systems," in Proceedings of the 5th ACM international conference on Distributed event-based system, pp. 113–124, ACM, 7 2011.

[22] Y. Chen, H. Huang, and W.-T. Tsai, "Scheduling simulation in a distributed wireless embedded system," SIMULATION, vol. 81, pp. 425–436, 6 2005.

[23] C. Ramachandran, S. Misra, and M. Obaidat, "On evaluating some agent-based intrusion detection schemes in mobile ad-hoc networks," Proceedings of the SPECTS 2007, pp. 594–601, 2007.

[24] A. Jain and C. Farkas, Ontology-Based Authorization Model for XML Data in Distributed Systems, pp. 57–82. IGI Global, 5 2010.

[25] Y. Wu, M. Zhao, A. Haeberlen, W. Zhou, and B. T. Loo, "Diagnosing missing events in distributed systems with negative provenance," ACM SIGCOMM Computer Communication Review, vol. 44, pp. 383–394, 8 2014.

[26] J. Mace, R. Roelke, and R. Fonseca, "Pivot tracing: dynamic causal monitoring for distributed systems," Communications of the ACM, vol. 63, pp. 94–102, 2 2020.

[27] J. Dale and D. Dwyer, "Calibration and display of distributed aperture sensor systems," in SPIE Proceedings, vol. 6559, pp. 65590G–65590G, SPIE, 4 2007.

[28] H. Chen, J. Yu, F. Liu, Z. Cai, and J. Xia, "Archipelago: A medical distributed storage system for interconnected health," IEEE Internet Computing, vol. 24, pp. 28–38, 3 2020.

[29] B. J. Garner and S. Kutti, "Application of dynamic knowledge in the design of distributed operating systems," in SPIE Proceedings, vol. 0786, pp. 391–397, SPIE, 5 1987.

[30] F. T. Brown, "A unified approach to the analysis of uniform one-dimensional distributed systems," Journal of Basic Engineering, vol. 89, pp. 423–432, 6 1967.

[31] S. M. Srinivasan, S. Mahbub, R. S. Sangwan, Y. Badr, and P. Mukherjee, Pattern Language for Designing Distributed AI Systems, pp. 467–477. Springer International Publishing, 12 2022.

[32] S. Lacroix, B. Eynard, J. L. Duigou, X. Godart, and C. Danjou, Distributed Approach for Integration in Industrial Systems, pp. 116–125. Germany: Springer Nature Switzerland, 2 2023.

[33] A. Campa and S. Gupta, "Synchronization in a system of kuramoto oscillators with distributed gaussian noise.," Physical review. E, vol. 108, pp. 064124–, 12 2023.

[34] J. Tan and X. Liu, "Distributed algorithm without iterations for an integrated energy system," Frontiers in Energy Research, vol. 10, 1 2023.

[35] Z. Xia, Y. Liu, D. Wang, and W. Gui, "Modified graph systems for distributed optimization," Science China Information Sciences, vol. 66, 11 2023.

[36] Y. Xiang, M. Pathan, G. Wei, and G. Fortino, "Availability, resilience, and fault tolerance of internet and distributed computing systems," Concurrency and Computation: Practice and Experience, vol. 27, pp. 2503–2505, 5 2015.

[37] A. V. Uzunov, E. B. Fernandez, and K. Falkner, "Assessing and improving the quality of security methodologies for distributed systems," Journal of Software: Evolution and Process, vol. 30, 8 2018.

[38] M. M. Mohammed, "Dynamic adaptation for distributed systems in model-driven engineering," in Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, pp. 146–151, ACM, 10 2022.

[39] V. Agate, A. D. Paola, G. L. Re, and M. Morana, DCAI - A Simulation Framework for Evaluating Distributed Reputation Management Systems, vol. 474, pp. 247–254. Springer International Publishing, 6 2016.

[40] D. J. Gulbransen, S. H. Black, A. C. Childs, C. L. Fletcher, S. M. Johnson, W. A. Radford, G. M. Venzor, J. P. Sienicki, A. D. Thompson, J. H. Griffith, A. A. Buell, M. F. Vilela, M. D. Newton, E. H. Takken, J. R. Waterman, and K. A. Krapels, "Wide-fov fpas for a shipboard distributed aperture system," in SPIE Proceedings, vol. 5406, pp. 305–316, SPIE, 8 2004.

[41] 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.

[42] A. Balogh and M. Krstic, "Advances in robust and nonlinear control of distributed parameter systems," International Journal of Robust and Nonlinear Control, vol. 16, pp. 731–732, 9 2006.

[43] Y. Yoon, S. Yoon, and M. Yoon, "Ubicomp - development of a distributed chemical event system," in Proceedings of the 2012 ACM Conference on Ubiquitous Computing, pp. 588–589, ACM, 9 2012.

[44] Q. Cai, J. Cao, G. Xu, and N. Zhu, "Distributed recommendation systems: Survey and research directions," ACM Transactions on Information Systems, vol. 43, pp. 1–38, 11 2024.

[45] J. Zhao, F. Dai, Y. Song, F. Wang, P. Zhang, and F. Li, A Distributed Optimal Formation Control for Multi-UGV System, pp. 567–575. Germany: Springer Nature Singapore, 9 2022.

[46] X. Fu and H. Cai, "Esec/sigsoft fse - a dynamic taint analyzer for distributed systems," in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 1115–1119, ACM, 8 2019.

[47] A. A. Lasode and W. F. Northrop, "Modeling energy recovery in (distributed) wastewater treatment systems," Sustainability in Environment, vol. 7, pp. p29–p29, 5 2022.

[48] J. Liang, S. Y. Ko, I. Gupta, and K. Nahrstedt, "Mon: management overlay networks for distributed systems," in Proceedings of the twentieth ACM symposium on Operating systems principles, pp. 1–2, ACM, 10 2005.

[49] G. Liu, T. Jiang, T. B. Ollis, X. Li, F. Li, and K. Tomsovic, "Resilient distribution system leveraging distributed generation and microgrids: a review," IET Energy Systems Integration, vol. 2, pp. 289–304, 12 2020.

[50] J. Dong, Y. Sun, R. Fu, C. Zhao, and M. Jiang, "Distributed precoder design for uplink tdd mu-mimo systems," in 2022 IEEE 8th International Conference on Computer and Communications (ICCC), vol. 2011, pp. 2429–2433, IEEE, 12 2022.

[51] N. A. Delessy, E. B. Fernandez, M. M. Larrondo-Petrie, and J. Wu, "Patterns for access control in distributed systems," in Proceedings of the 14th Conference on Pattern Languages of Programs, pp. 3–11, ACM, 9 2007.

[52] S. Goel and R. Buyya, Data Replication Strategies in Wide-Area Distributed Systems. IGI Global, 1 2011.

[53] D. Hughes, P. Greenwood, and G. Coulson, "A framework for testing distributed systems," in Proceedings. Fourth International Conference on Peer-to-Peer Computing, 2004. Proceedings., pp. 262–263, IEEE, 11 2004.

[54] H. Shu, Q. Liang, and J. Gao, "Distributed sensor networks deployment using fuzzy logic systems," International Journal of Wireless Information Networks, vol. 14, pp. 163–173, 3 2007.

[55] S. Wang, H. Yu, S. Yuan, S. E. Li, and Z. Ning, "Velocity planning for multi-vehicle systems via distributed optimization," in 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC), pp. 4491–4497, IEEE, 9 2023.

[56] L. Yan, L. Jiang, and Y. Xia, Distributed Fusion Estimation for Systems with Network Delays and Uncertainties, pp. 83–103. Springer Singapore, 11 2020.

[57] T. Osman and A. Bargiela, "Reliable distributed computing for decision support systems," Measurement and Control, vol. 32, pp. 115–118, 5 1999.

[58] D. Hatt, Moroccan Saints' Shrines as Systems of Distributed Knowledge, pp. 153–203. University of Calgary Press, 11 2009.

[59] S. Liu, H. Jiang, L. Zhang, and X. Mei, "Distributed adaptive optimization for generalized linear multiagent systems," Discrete Dynamics in Nature and Society, vol. 2019, pp. 1–10, 8 2019.

[60] 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.

[61] S. Whitlock, C. Scott, and S. Shenker, "Podc - brief announcement: techniques for programmatically troubleshooting distributed systems," in Proceedings of the 2013 ACM symposium on Principles of distributed computing, pp. 134–136, ACM, 7 2013.

[62] X. You, D. Wang, and J. Wang, Caching Optimization in Distributed MIMO Systems, pp. 109–137. Springer Singapore, 12 2020.

[63] S. Wang and C. Li, "Distributed stochastic algorithm for global optimization in networked system," Journal of Optimization Theory and Applications, vol. 179, pp. 1001–1007, 8 2018.

[64] W. Cao and Y. Xu, "Collocation method for optimal control of a fractional distributed system," Fractal and Fractional, vol. 6, pp. 594–594, 10 2022.

[65] B. P. Bhattarai, V. Chalishazar, D. J. Hammerstrom, and M. Maharjan, "Transactive energy systems for distributed blackstart and service recovery," IET Smart Grid, vol. 4, pp. 489–499, 3 2021.

[66] J.-W. Wang, H.-X. Li, and H.-N. Wu, "Distributed proportional plus second-order spatial derivative control for distributed parameter systems subject to spatiotemporal uncertainties," Nonlinear Dynamics, vol. 76, pp. 2041–2058, 2 2014.

[67] S. Muir, "Recursive in/formality: Time and ideology in a distributed monetary system," Anuac, vol. 6, pp. 77–83, 12 2017.

[68] K. D. Frampton, "Distributed group-based vibration control with a networked embedded system," Smart Materials and Structures, vol. 14, pp. 307–314, 1 2005.

[69] D. Peng, H. Yang, and B. Jiang, Distributed Switching Time Optimization for Interconnected Switched Systems, pp. 384–393. Germany: Springer Nature Singapore, 7 2022.

[70] D. Rutten, M. Zubeldia, and D. Mukherjee, "Distributed speed scaling in large-scale service systems," in Abstracts of the 2024 ACM SIGMETRICS/IFIP PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, pp. 95–96, ACM, 6 2024.

[71] P. Eaton, B. Anderson, S. Queen, I. Mackenzie, and D. Wynn, "Newdas - the newmont distributed ip data acquisition system," SEG Technical Program Expanded Abstracts 2010, pp. 1768–1772, 10 2010.

[72] J. Luo and Q. Wu, "Scan statistics with local vote for target detection in distributed system," EURASIP Journal on Advances in Signal Processing, vol. 2017, pp. 32–, 5 2017.

[73] I. C. Jong, "On stability of a circulatory system with weak distributed yielding effects," Journal of Applied Mechanics, vol. 37, pp. 870–873, 9 1970.

[74] A.-A.-R. Nayeem, M. Elshambakey, T. Dobbs, H. Lee, D. Crichton, Y. Zhu, C. Chokwitthaya, W. J. Tolone, and I. Cho, "A visual analytics framework for distributed data analysis systems," in 2021 IEEE International Conference on Big Data (Big Data), pp. 229–240, IEEE, 12 2021.

[75] W. Han, H. L. Trentelman, Z. Wang, and Y. Shen, "Distributed fault estimation for linear systems with actuator faults," International Journal of Robust and Nonlinear Control, vol. 30, pp. 6853–6878, 9 2020.

[76] W. Takase, Y. Matsumoto, A. Hasan, F. D. Lodovico, Y. Watase, and T. Sasaki, "Experience of a low-maintenance distributed data management system," Journal of Physics: Conference Series, vol. 513, pp. 032095–, 6 2014.

[77] Y. Li and Y. Shang, "Design and implementation of a distributed complex event processing system," in SPIE Proceedings, vol. 10322, pp. 103220S–, SPIE, 1 2017.

[78] R. van Engelen, M. Govindaraju, N. Koziris, and K. Psarris, "Session details: Distributed systems and grid computing (dsgc)," in Proceedings of the 2006 ACM symposium on Applied computing, ACM, 4 2006.

[79] R. Malik, C. Ramachandran, I. Gupta, and K. Nahrstedt, "Samera: a scalable and memory-efficient feature extraction algorithm for short 3d video segments.," in IMMERSCOM, p. 18, 2009.

[80] P. A. Orner and A. M. Foster, "A design procedure for a class of distributed parameter control systems," Journal of Dynamic Systems, Measurement, and Control, vol. 93, pp. 86–92, 6 1971.

[81] R. Lopes and D. A. Menascé, "A taxonomy of job scheduling on distributed computing systems," IEEE Transactions on Parallel and Distributed Systems, vol. 27, pp. 3412–3428, 12 2016.

[82] E. C. Foster and S. Godbole, Distributed Database Systems, pp. 361–370. Apress, 11 2016.