SurvivalRank: Decentralized Evolution of Search Models

Andrei Ionescu , Johan Pouwelse

EEMCS, Delft University of Technology, The Netherlands

A Master Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Master of Computer Science and Engineering
June 26, 2026

Name of the student: Andrei Ionescu
Final project course: CSE5000 Research Project
Thesis committee: Johan Pouwelse, Annibale Panichella

SurvivalRank: Decentralized Evolution of Search Models

Andrei Ionescu Johan Pouwelse
EEMCS, Delft University of Technology, The Netherlands

Abstract

Google has held roughly 90% of the global search market for over a decade [1], and the obstacles to challenging it have multiplied. In the framework of Aghion and Howitt [2], economic growth arises from creative destruction [3], where new innovations displace old ones, and the cycle repeats. But creative destruction requires open entry, and in search, entry is prohibitively costly. To compete, a new entrant must crawl billions of pages, accumulate years of interaction data, and build infrastructure capable of serving results in milliseconds. Large language models have not restored competitive entry, and with training costs growing at approximately 2.4× per year [4], they have consolidated it further. This paper proposes SurvivalRank, a decentralized architecture designed to restore the entry condition that Aghion and Howitt take as given but that current search markets lack. Rather than having a single entity design the ranking algorithm, the system allows anyone to contribute a ranking model, distributed via BitTorrent [5]. Peer-to-peer networks have previously demonstrated the feasibility of sharing and recommending content among strangers at scale [6]. Each peer runs a multi-armed bandit [7] that allocates queries across available models, shifting traffic toward those that perform better. Peers share evaluation evidence through a lightweight gossip protocol [8], [9]: every few seconds, each peer sends its latest observation for one random arm to one random stranger, using Lamport timestamps for deduplication. Models that fall sufficiently below the best performer are retired locally. When a superior model arrives, it displaces the incumbent and the cycle begins again. We evaluate SurvivalRank on five learning-to-rank benchmarks. A single peer converges to the oracle-best arm on every dataset, gossip cooperation closes a substantial fraction of the gap to the centralized oracle, and when a superior model is injected mid-run by a single peer, the network displaces the incumbent within 16–70 rounds after injection on all configurations. A Price's Equation [10] decomposition confirms that a discrete innovation effect, a sustained positive selection effect, and a destruction effect from arm retirement are all active simultaneously, the empirical signature of Schumpeterian creative destruction operating without central coordination.

1 Introduction

This paper presents SurvivalRank, a decentralized search system in which ranking models compete for traffic through multi-armed bandits, a family of algorithms that learn which option is best by trying each and observing the outcomes, and the best model emerges organically, without any central authority picking the winner. SurvivalRank is the first system to combine a fully decentralized architecture, a decentralized multi-armed bandit for traffic allocation, and gossip-based evidence sharing between peers, together enabling Darwinian competition between models submitted through permissionless open entry.

When Google launched in 1998, it rapidly displaced AltaVista, Lycos, and many other search engines, as users migrated to a demonstrably better ranking algorithm. That is how innovation is supposed to work: someone builds something better, users switch, and the old guard disappears. Economics has a name for this process. Schumpeter called it *creative destruction* [3], and Aghion and Howitt [2] formalized it into a Nobel Prize-winning model of economic growth, where an innovator captures the market and earns temporary monopoly rents, but other firms can invest in research, and when someone builds something better, the incumbent is displaced. Growth arises not from any single innovation, but from the continual repetition of this cycle.

That kind of disruption has not happened in general web search at global scale since. Google has held roughly 90% of the global search market for over a decade [1], and the obstacles to challenging it have multiplied. To compete, a new entrant would need to crawl and index billions of web pages, collect years of interaction data to train ranking models, build global infrastructure capable of serving results in milliseconds, and attract enough users for the feedback loop (more users, more data, better results) to begin working in its favour. No startup or university has succeeded in building a globally competitive alternative, and while some governments have sustained regional competitors, none has displaced Google internationally. The last notable attempt in global markets, Microsoft's Bing, required billions of dollars in sustained investment [11] and still holds a relatively small share of global search traffic.

More recently, large language models have entered the search market. ChatGPT, Perplexity, and Gemini now answer queries directly, ranking and synthesising sources within the model itself. This might appear to be the disruption that search has been waiting for, but it is not clear that it restores competitive entry. Training a frontier LLM requires investments of tens to hundreds of millions of dollars per training run, with costs growing at approximately 2.4× per year

and projected to exceed one billion dollars by 2027 [4]. The ranking of sources within responses is largely opaque, and no external contributor can meaningfully audit or improve the model's internal ranking process. Users may have traded one black box for another.

The key feature of the Aghion and Howitt model is that entry is open by design. In search, the barriers are structural. Size reinforces the incumbent, as more users bring more data, more data brings better results, and better results bring more users, which is an advantage that grows with scale and that the Aghion and Howitt model simply does not account for [12]. The result is that creative destruction, which in their framework emerges from open innovation races, does not operate in practice. Google's position is not merely a temporary monopoly rent awaiting displacement. It persists because the form of entry that Aghion and Howitt treat as structurally open is, in this domain, prohibitively costly.

2 Problem Description

The observation above suggests that rather than attempting to outcompete Google on its own terms, one could instead build a system that restores the conditions under which creative destruction can operate.

This paper focuses specifically on ranking: how a fixed set of candidate documents for a given query is ordered. Ranking is where the intelligence of a search engine is proven, and it is also where the feedback loop is strongest, as better ranking attracts more users, whose interactions generate more training data, which further improves ranking. If contributing a ranking model requires nothing more than implementing a function that takes a query and a set of candidate documents and returns a ranked list, then the cost of entry drops by orders of magnitude.

A student or a hobbyist with a laptop and access to a public learning-to-rank dataset could train a ranker in as little as an afternoon (for instance, by fine-tuning a gradient-boosted model on a query distribution the incumbent underserves, such as a specific language, domain, or document type), package it as a few megabytes of model weights, and submit it to a network of peers that are already serving search results. If the model ranks documents better than whatever is currently deployed, it should gradually receive more traffic. If it does not, it should be quietly retired. The entrant should not need to build crawling infrastructure, negotiate with a platform operator, or attract a user base. Entry should require nothing more than producing a better ranking function.

For this to work, two problems need to be solved. First, there must be a decentralized evaluation mechanism that can determine which model is best from local observations alone, without any single entity seeing all user interactions. Second, peers must be able to share what they have learned about model quality so that the network converges on the best model faster than any individual peer could on its own. Model distribution itself is addressed by existing peer-to-peer file sharing. A system that solves both research problems would restore the open entry condition that Aghion and Howitt [2] take as given, as the cost of attempting to displace the incumbent would be reduced to the cost of training a ranking model. Whether a decentralized system can actually achieve this is the central question of the paper.

Research Question and Contributions

The central research question decomposes into three parts. First, can peers independently identify better ranking models from local interaction signals? This asks whether a bandit, an algorithm that balances trying new models against exploiting the best known one, running on a single peer, seeing only its own queries, converges on the best available model. Second, can the network aggregate and propagate that information efficiently enough for learning to improve with scale? This asks whether gossip makes the network smarter than the sum of its isolated parts. Third, when a superior model appears, does it reliably displace weaker incumbents, producing sustained performance gains rather than short-lived fluctuations?

This paper makes three contributions. First, we present SurvivalRank, a decentralized multi-armed bandit algorithm for ranking-model selection with open entry, in which any participant can submit a new model and the network evaluates it without central coordination. Second, we demonstrate the viability of SurvivalRank through an operational open-source prototype. Third, we provide an experimental evaluation on five learning-to-rank benchmarks demonstrating single-peer convergence, gossip scaling, and creative destruction, with Price's Equation used to decompose aggregate performance change into selection, innovation, and destruction effects.

3 Related Work

The system proposed in this paper sits at the intersection of several mature literatures that have, until now, developed relatively independently. This section traces the relevant threads and identifies where they fall short of what our architecture requires. Table 1 provides a compact summary.

The multi-armed bandit problem captures a simple but fundamental tradeoff: should you exploit what you already know works, or explore something that might be better? When a single agent faces this problem, well-known algorithms like UCB1 [7] and Thompson Sampling [21] provide strong guarantees. The question relevant to this paper is what happens when many agents face a similar bandit problem simultaneously and can talk to each other.

Szörényi et al. [14] studied this in a peer-to-peer setting. In their model, every peer pulls one arm per round and then exchanges information with a few random neighbours. They showed that the probability of any individual peer playing a suboptimal arm decreases proportionally to $1/(Pt)$, where P is the number of peers and t is the number of rounds.

Subsequent work refined this idea along multiple dimensions. Zhu et al. [19] showed that the speed of learning depends not just on the number of peers but on how well connected they are. Their GossipUCB algorithm achieves better regret on densely connected networks than on sparse ones, with the precise relationship governed by the spectral properties of the communication graph. Lalitha and Goldsmith [18] extended the framework to Bayesian algorithms, proposing a decentralized version of Thompson Sampling that merges posterior beliefs across neighbours and achieves the same

Year	Paper	Decentralized Architecture	Decentralized Communication	Open Entry	Model Competition	Traffic Allocation
2013	Gossip Learning [13]	✓	✓	✓		
2013	Gossip Bandits [14]	✓	✓			✓
2017	Decentralized Personalisation [15]	✓	✓	✓		
2020	Contextual Model Selection [16]				✓	✓
2020	Insert-Eliminate [17]	✓	✓			✓
2021	Decentralized Thompson [18]	✓	✓			✓
2022	Federated Bandit [19]	✓	✓			✓
2024	FedPAE [20]	✓	✓	✓		
2026	**SurvivalRank**	✔	✔	✔	✔	✔

Table 1: Prior research domains that SurvivalRank combines.

asymptotic regret as a centralized agent that sees all observations directly. Chawla et al. [17] pushed in the direction of communication efficiency, designing an algorithm where agents exchange only the names of promising arms rather than full reward statistics, and proved that even this minimal communication is enough to reduce each agent's regret by a factor of P.

These results establish that gossip-based cooperation can make distributed bandits nearly as effective as centralized ones. However, all of these algorithms assume a fixed and known set of arms. They do not consider the possibility that new arms might appear during execution, which is essential for a system where anyone can contribute a new ranking model at any time.

More recently, Hu et al. [22] extended the decentralized bandit setting to adversarial environments, proposing DeMABAR, an algorithm that achieves near-optimal regret even when an adversary corrupts reward observations or when a fraction of agents behave in a Byzantine manner. Their trimmed-mean filtering mechanism limits the influence of corrupted neighbours on each agent's reward estimates. This threat model is directly relevant to our system, where a malicious peer could inflate or deflate its reported rewards to bias the network's model selection. However, DeMABAR assumes a fixed arm set known to all agents, synchronized communication epochs, and a homogeneous reward distribution, none of which hold in our asynchronous, open-entry setting, so our statistics channel remains exposed at the reward level to flooding and whitewashing. Beyond the integrity of reported statistics, the system is also exposed at the identity level, where Sybil [23] attacks let a single adversary fabricate many peer identities to amplify its influence.

A separate line of work has tackled the question of how to choose among competing models, but from a centralized perspective. The most directly relevant paper is CompeMLLM by Zhao et al. [24], which addresses multi-modal LLM workflows. Instead of assigning one model per subtask, CompeMLLM runs several models on the same subtask in parallel and uses an LLM to judge which result is best. This competitive approach consistently outperforms static assignment, demonstrating that model competition is a productive design principle.

The broader landscape also confirms this. Ong et al. [25] train router models to dynamically select between a stronger and a weaker LLM, halving inference cost without degrading quality. Chen et al. [26] cascade queries through progressively more expensive models, stopping once a confident answer is obtained. Chiang et al. [27] operate Chatbot Arena, where millions of pairwise human comparisons rank LLMs via Bradley-Terry ratings. Each of these systems relies on a central entity (a router, a judge, or a platform operator) that observes all feedback and determines which model wins.

Mueller et al. [20] partially decentralise through FedPAE, a peer-to-peer algorithm in which peers share heterogeneous models and construct weighted ensembles locally. FedPAE is decentralized and supports diverse architectures, but it blends models into ensembles rather than selecting among them. No model is ever displaced or retired on the basis of competitive evaluation.

Applying bandits to information retrieval is also not new. Radlinski et al. [28] learned diverse document rankings from click feedback using a bandit formulation. Yue and Joachims [29] introduced dueling bandits for online ranker evaluation, inferring preferences between ranking functions from interleaved comparisons. Schuth et al. [30] generalized this to multileaved comparisons, enabling the simultaneous evaluation of many rankers from a single set of user interactions. All of these methods assume a single-node setting, where one system observes all user interactions and uses that centralized feedback to update its ranking function. Our system lifts this evaluation process into a distributed network where no single peer sees all interactions.

Each of these papers solves part of the problem but not all of it. Decentralized bandits show that gossip-based cooperation approaches centralized performance, but they assume a fixed arm set. Centralized model competition demonstrates that competitive selection improves performance, but it requires a central judge.

Our contribution is a system that combines these elements: a gossip-augmented decentralized multi-armed bandit for ranking-model selection, with open entry of new models via BitTorrent.

4 Design of SurvivalRank

The system we propose is made up of four components: the *models* each peer can choose between, a *statistics-exchange* channel through which peers share what they have learned, a *leaderboard* that records each peer's current view of which model is best, and the *queries* each peer answers, against which models are judged. Figure 1 shows how these fit together within a peer and how peers relate to one another across the network.

The guiding principle is decentralisation: there is no central registry of models, no authoritative leaderboard, and no coordinator that tells peers what to do. Every peer executes the same logic independently, learns from its own experience, and improves that learning by exchanging evidence with whatever other peers it happens to meet. The architecture is what makes this possible, as the four components are deliberately decoupled so that a peer can join with any set of models, talk to any peer, and form its own opinion, without depending on any shared authority.

The models are the alternatives a peer chooses between when answering a query, being in our case ranking models that score and order candidate documents. Each peer holds its own set, and that set need not match any other peer's. A peer can join with whatever models it already has, and a new contributor can introduce a model at any time simply by adding it to its own set and letting the network discover it. No peer has authority over another peer's models, and no model is ever forced on the network. A model earns traffic only by performing well once peers begin trying it.

The statistics-exchange channel is how peers cooperate. Periodically, each peer tells one other peer how a particular model has been performing for it. Over time, these exchanges spread evidence about every model across the whole network, so that a peer benefits from the experience of others it has never directly interacted with.

Each peer maintains a leaderboard containing its current ranking of the available models by how well they have performed. The leaderboard is personal to each peer and it reflects that peer's own experience plus whatever evidence has reached it through statistics exchange, so two peers may disagree in the short term. Crucially, no peer can edit another's leaderboard directly, it can only contribute evidence that the other peer chooses how to incorporate.

Finally, the models need something to be judged on, namely the queries each peer serves, paired with feedback on how good the answer was. In principle these are the peer's own queries (through the use of click signals), so that each peer learns from the traffic it genuinely serves and the network as a whole reflects real and current demand. For our experiments in this thesis, we substitute a standard benchmark dataset, replayed as each peer's query stream, so that we can measure the system against a known ground truth. This substitution is purely an experimental convenience, as the rest of the system does not depend on where the queries come from. The query stream is a swappable input, so the same architecture supports both the controlled experiments here and an eventual fully decentralized deployment driven by live user traffic.

The flow within a peer is a short loop. A query arrives, the peer tries one of its models, observes how well it did, and updates its leaderboard accordingly. The flow between peers is a single channel, where statistics exchange carries learned evidence from one peer to another, feeding back into each recipient's loop. We present below the Algorithm 1 of our system, with its expanded version present in the Appendix. We have also attached a sample of the user-facing interface we have created for the live deployment, seen in Figure 2.

Algorithm 1 Gossip-Augmented Decentralized MAB

Require: Arm (model) set $\mathcal{K}$, identity i, queries-per-round Q, gossip interval T_g, observation TTL Δ_w
Operators: index φ, aggregator γ, update $\oplus$, eliminate ψ, prior s_0

Init: $\forall k \in \mathcal{K}$: $own_k \leftarrow s_0$; $\Pi_k \leftarrow \emptyset$; $\mathcal{K}^{\text{exc}} \leftarrow \emptyset$; $L \leftarrow 0$; $\Lambda \leftarrow \emptyset$

Async: on GOSSIPMSG$(s, k, stats, \ell, magnet)$:
  if $k \notin \mathcal{K}$: fetch via magnet; on completion $\mathcal{K} \leftarrow \mathcal{K} \cup \{k\}$; $own_k \leftarrow s_0$
  if $\ell \leq \Lambda[(s,k)]$: drop else: $\Lambda[(s,k)] \leftarrow \ell$; $\Pi_k[s] \leftarrow (stats,$ NOW$())$

Every T_g seconds: pick random $k \in \mathcal{K} \setminus \mathcal{K}^{\text{exc}}$, random peer j; $L \leftarrow L + 1$; send GOSSIPMSG$(i, k, own_k, L, magnet_k)$ to j
  evict $\Pi_k[s]$ where NOW$() - \Pi_k[s].$`last_received` $> \Delta_w$

for each query $q = 1, 2, \ldots$ **do**
  // Aggregate own + non-stale peer observations, select arm
  $\forall k \in \mathcal{K} \setminus \mathcal{K}^{\text{exc}}$: $\bar{c}_k \leftarrow \gamma\big(own_k, \{\Pi_k[s] : \text{non-stale}\}\big)$
  $k^* \leftarrow \arg\max_{k \in \mathcal{K} \setminus \mathcal{K}^{\text{exc}}} \varphi(\bar{c}_k)$
  // Observe reward and update local statistics
  $r \leftarrow$ REWARD(k^*, q); $own_{k^*} \leftarrow own_{k^*} \oplus r$
  $L \leftarrow L + 1$
  // Every Q queries: eliminate confidently dominated arms
  if $q \bmod Q = 0$ **then**
    $\mathcal{K}^{\text{exc}} \leftarrow \mathcal{K}^{\text{exc}} \cup \{k : \psi(\bar{c}_k, \{\bar{c}_{k'}\}_{k' \in \mathcal{K} \setminus \mathcal{K}^{\text{exc}}}) = \texttt{true}\}$
    $\mathcal{K} \leftarrow \mathcal{K} \setminus \mathcal{K}^{\text{exc}}$
  end if
end for

Gossip Protocol Semantics

This subsection specifies how gossip works in technical detail and why the design choices we made are necessary.

Each peer processes queries continuously as they arrive. A *round* boundary occurs every Q queries, at which point arm elimination is checked. Between round boundaries, the peer simply selects an arm, evaluates it, and updates its local statistics. The round counter serves only as a cadence for elimination and UI snapshots, not as a synchronisation barrier across peers. Gossip operates on its own wall-clock schedule, fully decoupled from the query loop.

Peers communicate through a single IPv8 [31] message type: GOSSIPMSG. Every T_g seconds, each peer picks one arm uniformly at random from its active set and one peer uniformly at random from IPv8's continuously-walked peer table, and sends a single tuple containing the arm identifier, the peer's own cumulative statistics for that arm (n, R or α, β), a

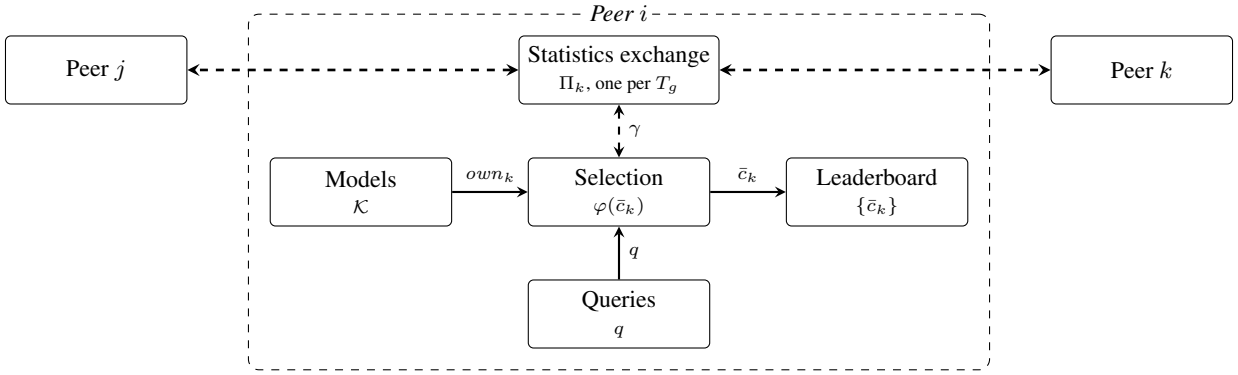


Figure 1: System architecture, with one peer shown in detail (dashed box) and two of its current gossip partners shown in simplified form. Each peer holds a local arm set $\mathcal{K}$, selects a model per query via φ over the aggregated statistics $\bar{c}_k = \gamma(own_k, \Pi_k)$, and maintains a leaderboard (the current $\{\bar{c}_k\}$ ranking). The statistics-exchange component sends one observation own_k per interval T_g to one random peer and folds incoming observations into Π_k; this is the only cross-peer interaction.

Lamport timestamp and a BitTorrent magnet link.

This design is deliberately minimal. A peer that gossips every 5 seconds reaches 720 distinct strangers per hour and over 120,000 per week. Spread is purely from each peer's own periodic emissions and no receiver ever re-forwards a message. This avoids the uncontrollable amplification of epidemic forwarding and provides natural rate control.

Each emission carries a Lamport stamp ℓ that is strictly increasing per sender. The sender increments its local Lamport counter L before every emission and before every local reward update, ensuring that newer observations always carry higher stamps. Receivers maintain a high-water mark $\Lambda[\langle s, k\rangle]$ for each $\langle$sender, arm$\rangle$ pair. An incoming tuple with $\ell \leq \Lambda[\langle s, k\rangle]$ is silently dropped, providing idempotency.

Each peer maintains, for every arm k, its own local evidence own_k and a table Π_k mapping each sender s to the latest gossip received from s about arm k. When a new GOSSIPMSG from sender s about arm k passes the Lamport gate, the receiver overwrites $\Pi_k[s]$ wholesale with the incoming statistics and records the current wall-clock time as `last_received`.

Entries in Π_k whose `last_received` is older than Δ_w seconds are excluded from aggregation. A peer that goes silent stops influencing decisions Δ_w seconds later. This ensures that the posterior reflects recent network-wide evidence regardless of how fast individual peers process queries.

Every GOSSIPMSG carries the arm's BitTorrent magnet link. A receiver that does not yet have arm k uses the magnet to start an asynchronous download. Once the download completes, the arm enters $\mathcal{K}$ and subsequent gossip for it contributes to the aggregate. There is no dedicated announcement message, as model discovery piggybacks on the same gossip that carries scoring information. This means discovery latency is $\Theta(T_g \cdot |\mathcal{K}|/|\mathcal{P}|)$ in expectation per neighbour, which is slower than a dedicated broadcast, but requiring no additional protocol machinery.

When a peer proposes a new model, it registers the arm locally with a random hex suffix (to prevent name collisions with concurrent proposals), begins seeding the model artifact and starts using the arm in its own bandit. The regular wall-clock gossip cycle picks up the new arm on rotation, the same as any other arm the peer is using. Other peers discover it when they receive a GOSSIPMSG for an arm they do not yet have.

When the elimination operator ψ determines that an arm k is confidently dominated, the arm is moved to a local exclusion set $\mathcal{K}^{\text{exc}}$ and removed from the active arm set $\mathcal{K}$. The exclusion set prevents the peer from re-adopting k when a stranger that has not yet eliminated it gossips statistics for k.

Elimination is strictly local, as no peer broadcasts its elimination decisions. This is a deliberate choice, since broadcasting eliminations would allow a malicious peer to force the network to drop a model by sending a fabricated message. By keeping elimination local, each peer must independently determine that an arm is confidently dominated based on its own aggregated evidence. Since peers receive similar observations over time through the gossip process, they should independently converge on the same elimination decisions.

5 Experiments and Performance Analysis

This section evaluates the system proposed in Section 4 through three experiments, each isolating one factor of the research question. The *Single-Peer Convergence Experiment* establishes that the bandit converges to the best ranking model in a single-peer setting. The *Multi-Peer Convergence with Gossip Experiment* measures how gossip-based cooperation affects convergence and quantifies the cost of decentralisation. The *Creative Destruction via Hot-Swap Experiment* tests whether creative destruction emerges when a superior model enters mid-run, using Price's Equation to decompose aggregate performance change.

All configurations are repeated over ten random seeds (see Appendix).

5.1 Datasets

We evaluate on five learning-to-rank datasets that span a range of sizes, feature sets, and provenance. **MQ2008** [32] is a benchmark from the LETOR 4.0 collection, containing 784 queries with 46 features per query-document pair, derived

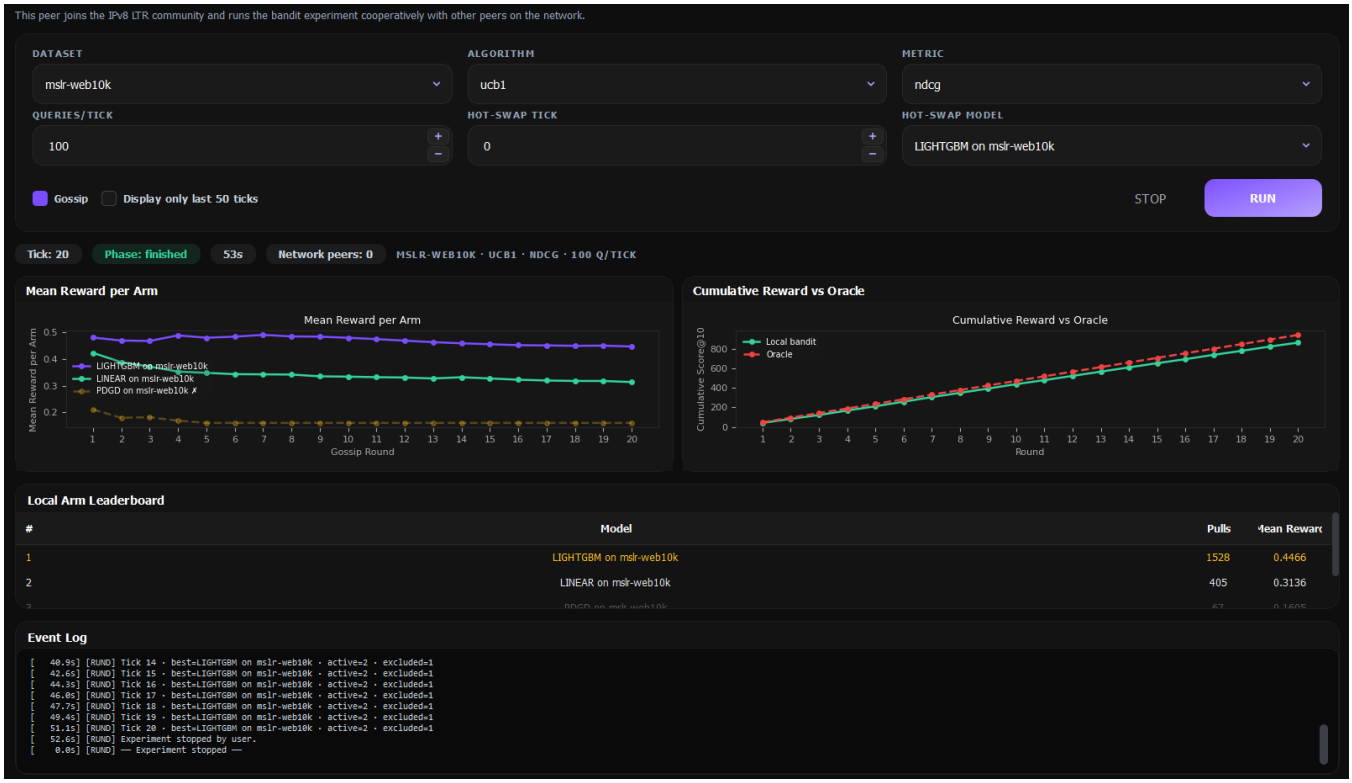


Figure 2: SurvivalRank UI showing the dashboard, model leaderboard, and live gossip status used to monitor decentralized model evolution.

from the TREC 2008 Million Query track over the Gov2 corpus. **MSLR-WEB10K** [33] contains 10,000 queries with 136 features, sampled from Microsoft Bing search logs. **MSLR-WEB30K** [33] is the larger variant of the same collection, containing 30,000 queries with the same 136 features. **Istella** [34] is a large-scale dataset released by the Istella search engine, containing 33,018 queries with 220 features and over 10 million query-document pairs. **AOL4FOLTR** [35] is a recent large-scale web search dataset containing 2.6 million queries from 10,000 users, specifically designed for federated and decentralized online learning-to-rank research.

Running across five datasets of varying scale and origin protects against dataset-specific artifacts and demonstrates the generality of the approach. MQ2008 provides coverage of a standard benchmark in the online learning-to-rank literature, MSLR-WEB10K and MSLR-WEB30K provide scale from a commercial search engine, Istella provides an independent commercial source, and AOL4FOLTR provides the most realistic setting for decentralized evaluation.

5.2 Ranking Models

Four ranking models serve as the bandit's arms. Three are available from the start across all experiments, while the fourth, XGBoost, is initially withheld and injected mid-run in the *Creative Destruction via Hot-Swap Experiment* to study how an active network responds to a superior late entrant.

LightGBM [36] is a gradient-boosted decision tree model trained offline on each dataset using LambdaRank loss with NDCG [37] (Normalized Discounted Cumulative Gain) as the target metric. It represents a strong, well-tuned baseline that the bandit should converge on quickly when no superior alternative is available. **LINEAR** is a simple linear scoring function whose weights are set by ridge regression on the training split. It represents a weak but stable model whose predictable performance anchors the lower end of the quality spectrum. **PDGD** [38] (Pairwise Differentiable Gradient Descent) is an online learning-to-rank model that learns its ranking function incrementally from click feedback, treating each click as a relative preference between documents and following an unbiased gradient estimate to improve the ranking. It represents an adaptive model with moderate performance that sits between LINEAR and LightGBM.

XGBoost [39] is a regularized gradient-boosted decision tree model, also trained offline with LambdaMART loss targeting NDCG. XGBoost uses a different regularisation strategy and tree construction algorithm than LightGBM, producing a model with comparable or superior ranking quality depending on the dataset.

The choice of four models spanning a range of expected quality, from weak (LINEAR) through moderate (PDGD) to strong (LightGBM) to potentially superior (XGBoost), enables meaningful evaluation of both convergence (can the system find the best model?) and creative destruction (can a late entrant displace an established incumbent?).

5.3 Evaluation Metrics and Baselines

The primary metric is **NDCG**. **NDCG** captures both the relevance and the position of documents in the ranked list, with

higher values indicating that more relevant documents appear earlier. The reward signal observed by the bandit at each query is **NDCG@10** of the ranking produced by the selected model, normalized to $[0, 1]$.

Our baseline is the centralized oracle, an evaluator that always selects the model with the highest expected **NDCG** for each query, calculated from precomputed metric scores across all models, representing the performance ceiling of a system with perfect information and no exploration cost.

We report convergence and cooperation through three derived metrics. **Cumulative regret** is $\mathrm{regret}(t) = \mathrm{oracle}(t) - \mathrm{algo}(t)$, the running difference between the oracle's cumulative reward and the algorithm's, with a flattening regret curve signaling convergence. The **gap to oracle** is this same difference at the final round, reported per peer, and measures the cost of decentralized learning. The **best-arm identification rate** is the fraction of $\langle \text{peer}, \text{run} \rangle$ pairs whose final highest-mean arm matches the dataset-best arm.

5.4 Single-Peer Convergence Experiment

In this experiment we prove that peers in our system can individually converge to the best available model through scoring only their own queries.

Hypothesis: Peers using Thompson Sampling or UCB1 converge to the best model (XGBoost), with a small steady-state gap to the oracle attributable to residual exploration.

Setup: A single peer runs Algorithm 1 over the four locally available models (LightGBM, XGBoost, PDGD, Linear) with no gossip and no model injection. We compare both algorithms against each other, running $T = 100$ rounds of $Q = 100$ queries each ($10{,}000$ pulls per run) on all five datasets. Every $\langle \text{dataset}, \text{algorithm} \rangle$ configuration follows the shared repetition and statistical reporting procedure.

Results: Figure 3 plots the cumulative regret across all five datasets. On 4 of the 5 datasets, regret growth flattens within the first few tens of rounds and remains approximately linear thereafter at a small slope, consistent with the expected residual-exploration term. The uncertainty bands remain narrow, indicating stable convergence behaviour.

Table 2 reports, for each $\langle \text{dataset}, \text{algorithm} \rangle$, the modal best-arm choice across the ten seeds, alongside the highest-mean arm, calculated directly from the precomputed per-query metric scores. Both algorithms identify the dataset-best arm on all five datasets, unanimously on MSLR-WEB10K and AOL4FOLTR ($10/10$ seeds) and on a clear majority elsewhere. The closest configurations are MQ2008 and the two MSLR datasets, where small gaps between the top two arms (e.g. mean reward for XGBoost at 0.492 vs LightGBM at 0.477 on MQ2008) make UCB1's Hoeffding bounds slow to separate them within $10{,}000$ queries. Thompson Sampling, which exploits posterior uncertainty more aggressively, reaches the highest-mean arm on a comparable or larger fraction of seeds in these tighter cases.

Figure 7 in the Appendix shows the final pull distribution for each configuration. Both algorithms concentrate pulls on the highest-mean arm on most datasets, but Thompson Sampling consistently allocates a larger fraction to the modal best arm than UCB1, reflecting its more aggressive exploitation under low residual uncertainty.

Dataset	Algorithm	Modal best arm	Mode share	Dataset-best arm
MSLR-WEB10K	UCB1	XGBoost	$10/10$	XGBoost
MSLR-WEB10K	Thompson	XGBoost	$10/10$	XGBoost
MSLR-WEB30K	UCB1	XGBoost	$6/10$	XGBoost
MSLR-WEB30K	Thompson	XGBoost	$6/10$	XGBoost
Istella	UCB1	XGBoost	$6/10$	XGBoost
Istella	Thompson	XGBoost	$6/10$	XGBoost
MQ2008	UCB1	XGBoost	$7/10$	XGBoost
MQ2008	Thompson	XGBoost	$8/10$	XGBoost
AOL4FOLTR	UCB1	XGBoost	$10/10$	XGBoost
AOL4FOLTR	Thompson	XGBoost	$10/10$	XGBoost

Table 2: Best-arm identification per $\langle \text{dataset}, \text{algorithm} \rangle$. "Mode share" is the fraction of seeds whose final highest-mean arm matches the modal choice. " Dataset-best arm" is the arm with the highest precomputed mean NDCG@10 over the dataset's test queries.

Both UCB1 and Thompson Sampling converge to the highest-mean arm on every dataset. Thompson consistently gets more cumulative reward than UCB1 across the entire suite, reflecting Thompson's faster posterior collapse around the identified winner. The hypothesis of convergence with a residual-exploration gap holds, since the empirical leader is XGBoost on every dataset in the suite.

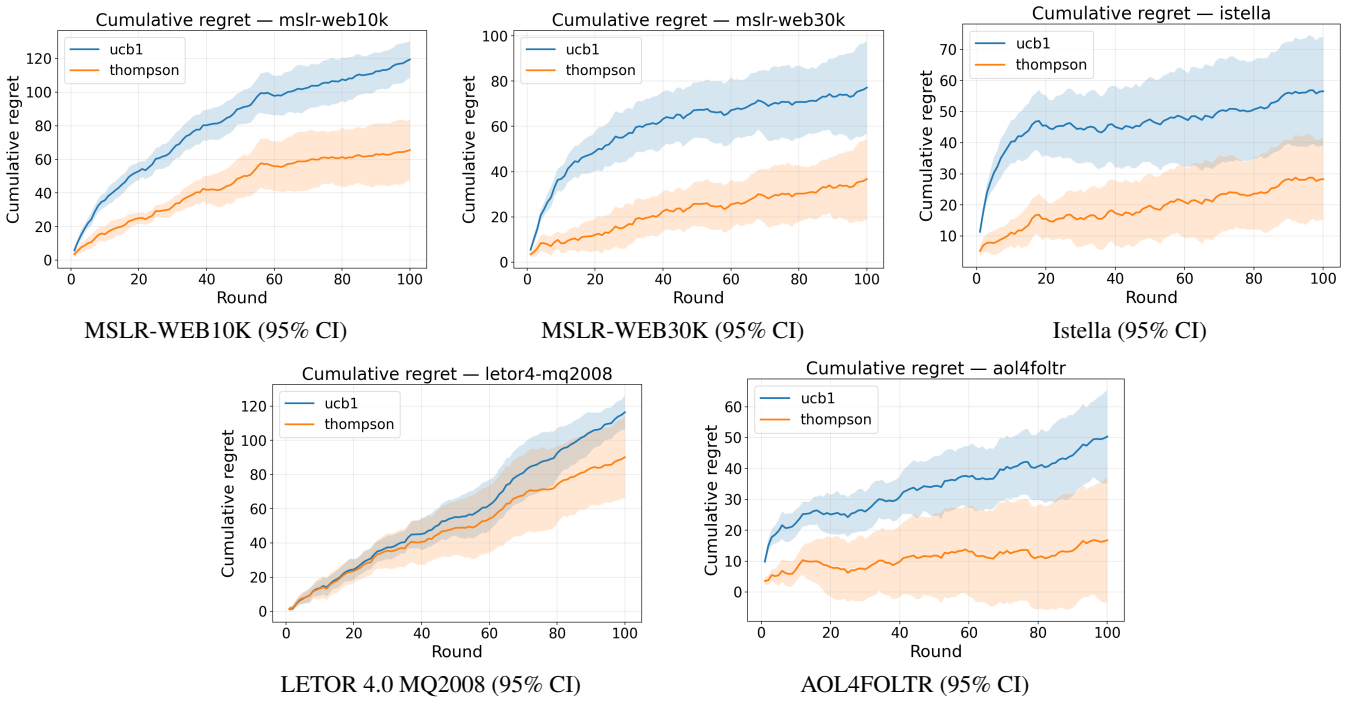
5.5 Multi-Peer Convergence with Gossip Experiment

In this experiment we prove that within our system, without any global coordination, each peer converges to the best model faster by cooperation.

Hypothesis: With gossip enabled, adding more peers should improve convergence because each peer benefits from the others' observations. The practical question is how large the gossip benefit is relative to the oracle upper bound.

Setup: Each peer runs Algorithm 1 over the same four models as the *Single-Peer Convergence Experiment* with no model injection. We vary the network size $N \in \{5, 10, 20\}$ with gossip enabled, and also run the $\langle N{=}20, \text{gossip off} \rangle$ case where 20 peers each run independently with no inter-peer communication. Every peer processes $Q{=}100$ queries per round for $T{=}100$ rounds ($10{,}000$ queries per peer). We use UCB1 as the bandit algorithm, as the *Single-Peer Convergence Experiment* established that Thompson Sampling has the better single-peer baseline. Thus, we choose UCB1 here as the conservative test of cooperation, since UCB1's deterministic exploration term keeps it pulling worse arms for longer than Thompson's posterior-driven exploitation. The cooperation benefit measured under UCB1 is therefore a lower bound on what gossip can deliver.

Results: Figure 4 compares three trajectories per dataset: the oracle (network upper bound), gossip-on with $N{=}20$ (our system), and gossip-off with $N{=}20$ (independent peers, no communication). The gap between gossip-on and oracle is the cost of decentralized learning under our protocol, while the gap between gossip-off and gossip-on is the measured value of cooperation. On every dataset, gossip-on closes a substantial fraction of the gossip-off gap to oracle. On MSLR-WEB10K, the gap falls from 84.7 to 50.1 cumulative NDCG@10, and on MSLR-WEB30K it falls from 77.3



MSLR-WEB10K (95% CI) MSLR-WEB30K (95% CI) Istella (95% CI)

LETOR 4.0 MQ2008 (95% CI) AOL4FOLTR (95% CI)

Figure 3: Cumulative regret of UCB1 and Thompson Sampling against the oracle-best arm on all five datasets. On 4 of the 5 datasets, regret growth flattens within the first few tens of rounds, and the steady-state slope is small relative to the per-round oracle reward, confirming convergence with a residual-exploration gap.

to 56.1. The improvement is similar on the remaining three datasets.

Table 3 reports the final-round per-peer-average cumulative reward, the gap to oracle, and the best-arm identification rate. The gap-to-oracle is largely flat across $N \in \{5, 10, 20\}$ on every dataset, as most of the cooperation benefit is already captured at $N=5$ (e.g. AOL4FOLTR goes from 37.6 at $N=5$ to 39.3 at $N=20$, and MSLR-WEB10K stays near 50 across all three sizes). Each additional peer beyond a small swarm seemingly contributes progressively less new information about arms the swarm already characterises well, and the uncertainty bands tighten as N grows while the means barely move.

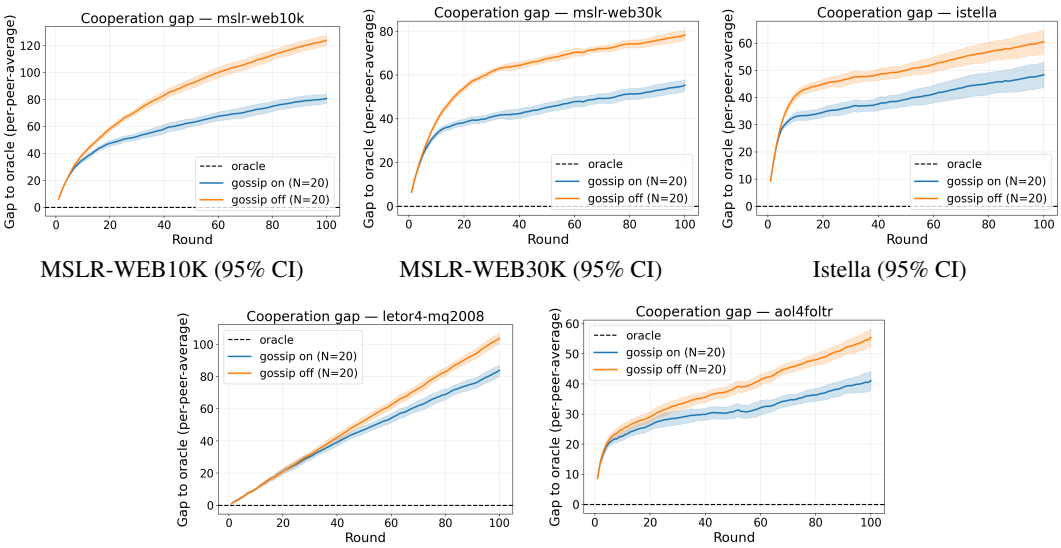
Best-arm identification follows a similar pattern. With gossip-on the rate is $\geq 91.5\%$ at $N=20$ on every dataset and reaches 100% on three of the five and with gossip off it drops to 73% - 100%. This angle measures the benefit of cooperation based on the algorithm's final state rather than cumulative reward, as when operating in isolation, peers occasionally converge on a suboptimal arm, whereas shared evidence enables them to more reliably reach the same correct conclusion.

On every dataset in the suite, gossip provides a measurable and statistically significant improvement over the no-communication baseline. The relative magnitude varies with the reward-gap structure of the dataset, as the largest gain appears to be on the noisier MSLR datasets where independent peers seem to struggle most.

Gossip turns a collection of independently-learning peers into a faster-converging swarm. The benefit shows up in both the cumulative reward (smaller gap to oracle) and the end-state (high best-arm identification at $N=20$). The gain saturates beyond $N \approx 5$ peers, suggesting that small-to-medium swarms already capture most of the cooperation benefit, and that increasing the network further yields diminishing returns under fixed per-peer query budgets.

Dataset	Cond.	N	Final reward	Gap to oracle	Best-arm rate
MSLR-WEB10K	gossip on	5	5078.0 ± 5.4	51.9 ± 5.4	100.0%
	gossip on	10	5079.7 ± 4.5	50.2 ± 4.5	100.0%
	gossip on	20	5079.7 ± 1.8	50.1 ± 1.8	100.0%
	gossip off	20	5045.2 ± 2.6	84.7 ± 2.6	100.0%
MSLR-WEB30K	gossip on	5	4840.6 ± 5.7	55.8 ± 5.7	80.0%
	gossip on	10	4842.1 ± 2.8	54.3 ± 2.8	95.0%
	gossip on	20	4840.3 ± 2.2	56.1 ± 2.2	91.5%
	gossip off	20	4819.1 ± 2.5	77.3 ± 2.5	73.0%
Istella	gossip on	5	6972.1 ± 5.0	49.4 ± 5.0	82.0%
	gossip on	10	6971.9 ± 3.8	49.6 ± 3.8	100.0%
	gossip on	20	6972.6 ± 3.2	48.9 ± 3.2	100.0%
	gossip off	20	6958.0 ± 2.6	63.5 ± 2.6	74.0%
MQ2008	gossip on	5	4825.6 ± 4.4	92.1 ± 4.4	100.0%
	gossip on	10	4830.9 ± 4.5	86.8 ± 4.5	100.0%
	gossip on	20	4828.5 ± 4.1	89.2 ± 4.1	98.5%
	gossip off	20	4811.2 ± 4.0	106.5 ± 4.0	83.0%
AOL4FOLTR	gossip on	5	8923.4 ± 4.7	37.6 ± 4.7	100.0%
	gossip on	10	8924.5 ± 2.9	36.6 ± 2.9	100.0%
	gossip on	20	8921.8 ± 2.9	39.3 ± 2.9	100.0%
	gossip off	20	8907.8 ± 3.1	53.3 ± 3.1	100.0%

Table 3: Final-round per-peer-average cumulative reward and gap to oracle for every configuration. The best-arm rate is the fraction of $\langle$peer, run$\rangle$ pairs whose final highest-mean arm matches the dataset-best model.



MSLR-WEB10K (95% CI) MSLR-WEB30K (95% CI) Istella (95% CI)

LETOR 4.0 MQ2008 (95% CI) AOL4FOLTR (95% CI)

Figure 4: Gap to oracle (per-peer-average cumulative NDCG@10) at $N{=}20$: oracle (dashed line at zero) vs gossip-on (blue) vs gossip-off (orange). Lower is better. The height of each coloured curve is the cost of decentralisation under that condition and the vertical separation between gossip-off and gossip-on is the value of cooperation.

5.6 Creative Destruction via Hot-Swap Experiment

In this experiment we test whether a superior model, which is injected mid-run into a network that has already converged on an incumbent, reliably displaces it through independent local bandit decisions and gossip propagation.

Hypothesis: The bandit displaces the incumbent and shifts network-wide traffic to XGBoost on every dataset, as per Table 2. The displacement should be visible as a crossover in traffic share, with LightGBM's share falling while XGBoost's rises.

Setup: The network starts with $N{=}20$ peers, all running UCB1 over only three of the four models: LightGBM, PDGD, and LINEAR. XGBoost is withheld. The bandit converges on the best model in the reduced pool in early rounds (LightGBM, as shown by the *Single-Peer Convergence Experiment*). At a configured injection round t_{inject}, a single designated peer proposes XGBoost. It registers the arm locally, begins seeding the model artifact, and starts using it in its own bandit. There is no immediate broadcast, as the regular wall-clock gossip cycle picks up XGBoost on rotation, and other peers discover it when they receive a GossipMsg for an arm they do not yet have. We run with $t_{\text{inject}} \in \{10, 25, 50\}$ over $T{=}200$ rounds ($Q{=}100$ queries per round, 20,000 queries per peer), on all five datasets, following the shared experimental protocol. The horizon is doubled relative to the *Single-Peer Convergence Experiment* and the *Multi-Peer Convergence with Gossip Experiment* to give post-injection convergence enough budget to mature.

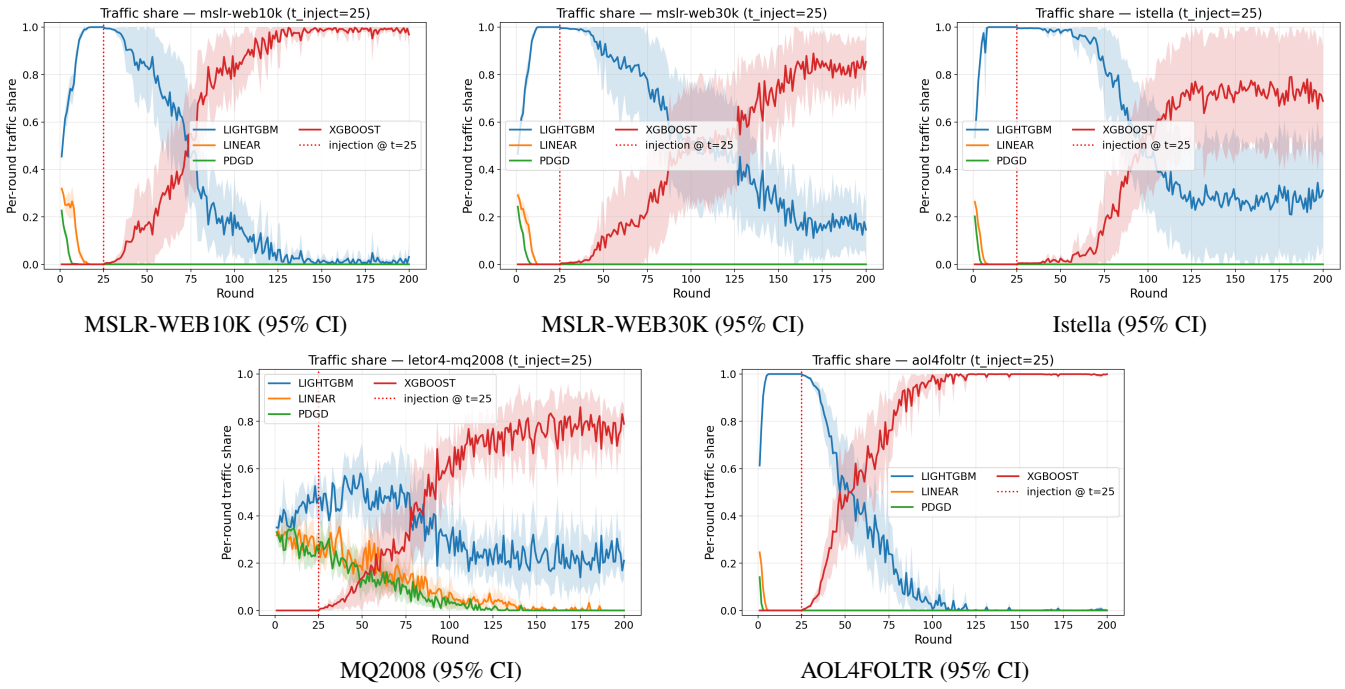
We report three quantities per $\langle \text{dataset}, t_{\text{inject}} \rangle$:

1. **Plurality round:** The first round after injection at which XGBoost holds the largest per-round traffic share of any model in the active pool, measuring how quickly the bandit reallocates traffic so that the superior model is the most-used arm network-wide
2. **Final XGBoost share:** The fraction of network-wide pulls in the final round allocated to XGBoost, capturing the post-displacement steady-state distribution
3. **Best-arm identification rate:** The fraction of $\langle \text{peer}, \text{seed} \rangle$ pairs whose final highest-mean arm is XGBoost

Results: Table 4 reports the three metrics for every configuration. Two observations are worth remarking.

Firstly, XGBoost becomes the plurality arm on $10/10$ seeds in every configuration. The plurality lag, the number of rounds between injection and the round at which XGBoost overtakes every other arm in network-wide share, ranges from $\sim$ 16 rounds (AOL4FOLTR at $t_{\text{inject}}{=}10$) to $\sim$ 70 rounds (MSLR-WEB30K at $t_{\text{inject}}{=}25$). The lag is broadly comparable across the three injection points within each dataset, so entry timing shifts the absolute plurality round but does not structurally penalise late entrants.

Secondly, XGBoost network share consistently stabilises as the plurality and frequently as the outright majority. Final-round shares range from 52% (MSLR-WEB30K at $t_{\text{inject}}{=}10$) up to 100% (AOL4FOLTR and MSLR-WEB10K at $t_{\text{inject}}{=}50$), with best-arm identification at 82.5–100% across every configuration. On datasets where the XGBoost–LightGBM gap is small (e.g. MSLR-WEB30K), the share oscillates in the 50–85% band rather than collapsing all traffic onto XGBoost, while on datasets with a large gap (AOL4FOLTR), the swarm allocates almost all of its queries to XGBoost by the end of the run.



MSLR-WEB10K (95% CI) MSLR-WEB30K (95% CI) Istella (95% CI)

MQ2008 (95% CI) AOL4FOLTR (95% CI)

Figure 5: Per-round traffic share of each model over rounds at $t_{\text{inject}}=25$ (marked as a dashed vertical line) for every dataset. XGBoost enters at injection and rises gradually past LightGBM over the ensuing rounds, eventually stabilising as the plurality arm. Shaded bands mark the 95% confidence-interval.

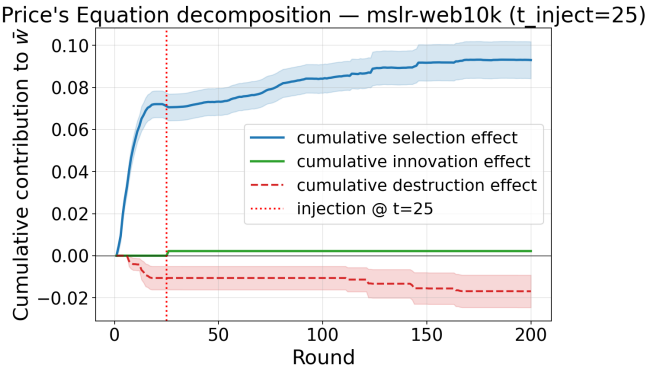
Dataset	t_{inject}	Plur. round	Final XGB share	Best-arm rate
MSLR-WEB10K	10	25.8 ± 1.0	81.3% ± 12.1%	100.0%
	25	69.2 ± 12.5	96.9% ± 5.2%	99.0%
	50	79.7 ± 5.2	100.0% ± 0.0%	100.0%
MSLR-WEB30K	10	42.9 ± 8.9	52.0% ± 10.2%	99.0%
	25	95.0 ± 20.4	85.4% ± 11.1%	93.5%
	50	84.2 ± 3.4	78.0% ± 11.7%	97.0%
Istella	10	54.2 ± 3.1	60.5% ± 9.9%	100.0%
	25	88.1 ± 8.7	68.9% ± 23.3%	82.5%
	50	101.7 ± 4.7	96.7% ± 6.5%	100.0%
MQ2008	10	30.5 ± 3.0	72.3% ± 8.6%	100.0%
	25	69.7 ± 10.5	78.9% ± 7.6%	100.0%
	50	87.5 ± 12.2	76.0% ± 11.0%	100.0%
AOL4FOLTR	10	25.6 ± 1.1	100.0% ± 0.0%	100.0%
	25	51.5 ± 7.7	99.9% ± 0.1%	99.5%
	50	73.6 ± 2.9	99.9% ± 0.1%	100.0%

Table 4: Creative-destruction metrics for every ⟨dataset, t_{inject}⟩ configuration at $T=200$. The plurality round is reported as an absolute round index (not an offset from t_{inject}), and is the first round after injection at which XGBoost holds the largest per-round share of any model in the active pool. Final XGBoost share is the fraction of network-wide pulls allocated to XGBoost in the last round, and best-arm rate is the fraction of ⟨peer, seed⟩ pairs whose final highest-mean NDCG@10 arm is XGBoost.

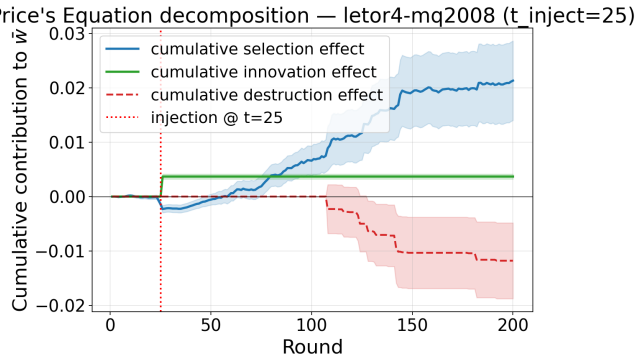
Figure 5 shows the per-round traffic share of each model on each dataset at the median injection round $t_{\text{inject}}=25$. The pattern is consistent across every dataset, as before injection LightGBM dominates ($\sim$ 70–80% share, with PDGD and LINEAR splitting the rest). At the injection point XGBoost's share rises gradually as the one-emission-per-round gossip protocol propagates the new arm peer-by-peer and each newly-aware peer's bandit allocates exploration pulls to it. XGBoost overtakes LightGBM as the plurality arm within roughly 26–70 rounds across datasets. After the crossover, the XGBoost share stabilises in the 50–100% band and the LightGBM share shrinks to the residual.

To formally test whether creative destruction is operating, we apply Price's Equation [10] to decompose the round-by-round change in network-average reward. It was originally formulated in evolutionary biology to decompose the change in the average value of a trait across generations into a component due to differential selection and a component due to transmission bias. Holm [40] later adapted it to industrial dynamics, decomposing aggregate productivity change across firms into a selection effect (market share reallocation toward more productive firms), an innovation effect (entry of new firms), and a destruction effect (exit of old firms).

In our setting, the population is the set of ranking models active in round t, the weight $w_k(t)$ is the fraction of network-wide pulls allocated to model k in that round, and the fitness f_k is the model's mean reward (precomputed NDCG@10 over the test queries). Because f_k is a fixed property of each model rather than a changing estimate, the decomposition cleanly separates three sources of change in the network-



MSLR-WEB10K (95% CI)



MQ2008 (95% CI)

Figure 6: Price's Equation decomposition at t_{inject}=25 for two representative datasets. The innovation term (green) spikes at the injection round as XGBoost enters with a higher mean reward than the incumbent pool; the selection term (blue) remains positive for many rounds afterwards as the bandit reallocates traffic to XGBoost; and the destruction term (red, dashed) contributes small positive steps as dominated arms are retired.

average reward $\bar{w}(t) = \sum_k w_k(t)\, f_k$:

$$\begin{aligned}\Delta\bar{w}(t) = &\underbrace{\sum_{k\in\mathcal{K}_{t-1}\cap\mathcal{K}_t} \Delta w_k(t) f_k}_{\text{selection effect}} \\ &+ \underbrace{\sum_{k\in\mathcal{K}_t\setminus\mathcal{K}_{t-1}} w_k(t) f_k}_{\text{innovation effect}} \\ &- \underbrace{\sum_{k\in\mathcal{K}_{t-1}\setminus\mathcal{K}_t} w_k(t-1) f_k}_{\text{destruction effect}}.\end{aligned} \tag{1}$$

where $\Delta w_k(t) = w_k(t) - w_k(t-1)$ is the change in traffic share for model k, and $\mathcal{K}_t$ denotes the active arm set at round t.

The *selection effect* captures reallocation among models that persist across both rounds. It is positive when the bandit shifts traffic from lower-performing to higher-performing models, as the Δw_k terms are positive for arms with high f_k and negative for arms with low f_k, so the weighted sum is positive whenever reallocation favours quality.

The *innovation effect* captures the contribution of models that enter at round t. It is positive when a newly arriving model has a high f_k and receives nonzero traffic. At the injection round, this term spikes as XGBoost enters with a higher mean than the incumbent pool.

The *destruction effect* captures the removal of models that are eliminated between rounds $t-1$ and t. It is subtracted when a low-f_k model is eliminated, removing its (small but nonzero) traffic share raises the population-weighted average, producing a positive net contribution to $\Delta\bar{w}$. This term is nonzero only when the elimination operator ψ fires.

If the selection and innovation effects are positive and sustained in the rounds following injection, and the destruction effect removes low-fitness arms, creative destruction is operating. If only the selection effect is present and the innovation effect is zero, entry may be formally open yet practically ineffective. Figure 6 plots the selection, innovation, and destruction components over rounds for two representative datasets at t_{inject}=25. Three characteristic features are visible on every dataset:

- The innovation effect is zero everywhere except the injection round, where it jumps upward as XGBoost enters the pool with mean reward above the pre-injection best. This is the innovation event in Holm's [40] adaptation of Price's Equation: a discrete, one-shot jump in the performance frontier driven by a new entrant. In Figure 6 this appears as a single tall bar at t=25 on both panels.
- For many rounds after injection the selection effect remains positive as the bandit gradually shifts traffic from LightGBM to XGBoost. The integral of the post-injection selection curve is the dominant component of the cumulative-reward improvement attributable to the swarm's growing recognition of the new arm's superiority.
- The destruction effect becomes nonzero when the elimination operator ψ fires and retires a dominated arm. Its removal raises the population-weighted average by stripping out the traffic share of a low-fitness model, producing a small but positive contribution to $\Delta\bar{w}$. In Figure 6 this appears as a modest upward step at the round where LINEAR or PDGD is eliminated.

The system reliably elevates the superior new model to plurality of network-wide traffic on every $\langle$dataset, $t_{\text{inject}}\rangle$ configuration we evaluate, with discovery completed within a few to a few tens of rounds and plurality attained within roughly 16–70 rounds after injection. The Price's Equation decomposition confirms that a discrete innovation event, a sustained positive selection effect, and a destruction effect from arm retirement are all operating, the empirical signature of the creative-destruction process Aghion and Howitt describe.

The penalty of late entry shows up not in whether plurality is attained (it always is), but in the absolute round at which

the swarm internalises the new arm. With $T=200$ rounds, final-round XGBoost shares are 52–100% and best-arm identification rates are 82.5–100%, comparable to the *Multi-Peer Convergence with Gossip* experiment's results.

6 Conclusions and Future Directions

This paper set out to ask whether creative destruction can operate in search without central coordination. The answer, across five learning-to-rank benchmarks and three experiments, appears to be yes. A single peer running Thompson Sampling or UCB1 converges to the highest-mean arm on every dataset. Gossip cooperation closes a substantial fraction of the gap to the centralized oracle, with most of the benefit already captured at five peers. And when a superior model is injected mid-run by a single contributor, the network elevates it to the plurality arm within 16–70 rounds after injection on every configuration tested, without any authority announcing the transition or mandating the switch.

The Price's Equation decomposition confirms that all three components of Holm's adaptation are simultaneously active following injection, marked by a discrete innovation spike as XGBoost enters the pool above the incumbent frontier, a sustained positive selection effect as the bandit gradually shifts traffic toward the superior entrant, and a destruction effect as dominated arms are retired and stripped of their traffic share. This is the empirical signature of Schumpeterian creative destruction, and it emerges here from nothing more than local bandit decisions and a lightweight gossip protocol.

The system is not without its limitations, however. The experiments evaluated here assume a model pool in which one arm is unambiguously superior across the dataset as a whole. This is a convenient property for demonstrating convergence, but it is not guaranteed in practice, as a new entrant might outperform the incumbent on one query distribution while underperforming on another, and it remains an open question how the bandit and gossip protocol behave when no single arm dominates. Whether the system reaches a stable mixed-traffic equilibrium, cycles between arms, or converges slowly on a suboptimal choice under such conditions is a natural target for future experimental work.

Another direction concerns the fidelity of the evaluation signal. The experiments substitute benchmark datasets for live click feedback, replaying precomputed relevance judgements as the reward signal in place of genuine user interactions. Click feedback is noisier, position-biased, and user-dependent in ways that offline NDCG scores are not, and the behaviour of the bandit and gossip protocol under that noisier signal remains to be established.

The gossip protocol does not prevent a peer from lying about its own observations, and a malicious peer can inflate or deflate the rewards it reports for any arm, biasing the network's belief about model quality without detection. At the identity level, a Sybil attacker [23] can register many peer identities and coordinate their false reports to amplify this bias further. Protecting the system against both classes of threat is a promising avenue for future work. For statistical robustness, the trimmed-mean filtering mechanism of Hu et al.'s [22] DeMABAR is a natural starting point, as it limits the influence of corrupted observations on each agent's reward estimates. For identity robustness, two candidates are MeritRank [41], a Sybil-tolerant reputation system that weights peer contributions by merit rather than identity count, and the attestation-revocation mechanism of Chotkan et al. [42], which provides a way to invalidate compromised peer identities across a distributed network without requiring a central authority.

To conclude, SurvivalRank demonstrates that the open entry condition Aghion and Howitt treat as given can be reengineered into search at the protocol level. The cost of attempting to displace the incumbent is reduced to the cost of training a ranking model. Whether that is sufficient to restore competitive dynamics in practice depends on adoption, and adoption is not a technical problem. But the technical barrier, which this paper identifies as the absence of a decentralized evaluation and propagation mechanism, has been removed.

References

[1] StatCounter, *Search engine market share worldwide*, https://gs.statcounter.com/search-engine-market-share, Accessed March 2026, 2025.

[2] P. Aghion and P. Howitt, "A model of growth through creative destruction," *Econometrica*, vol. 60, no. 2, pp. 323–351, 1992. DOI: 10.2307/2951599

[3] J. A. Schumpeter, *Capitalism, Socialism and Democracy*. New York: Harper and Brothers, 1942.

[4] B. Cottier, R. Rahman, L. Fattorini, N. Maslej, T. Besiroglu, and D. Owen, *The rising costs of training frontier AI models*, arXiv:2405.21015, 2024.

[5] B. Cohen, "Incentives build robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer Systems*, vol. 6, 2003, pp. 68–72.

[6] J. Wang, M. J. Reinders, J. Pouwelse, and R. L. Lagendijk, "Wi-fi walkman: a wireless handheld that shares and recommends music on peer-to-peer networks," in *Embedded Processors for Multimedia and Communications II*, S. Sudharsanan, V. M. B. Jr., and S. Panchanathan, Eds., International Society for Optics and Photonics, vol. 5683, SPIE, 2005, pp. 155–163. DOI: 10.1117/12.588509 [Online]. Available: https://doi.org/10.1117/12.588509

[7] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2–3, pp. 235–256, 2002. DOI: 10.1023/A:1013689704352

[8] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003, pp. 482–491. DOI: 10.1109/SFCS.2003.1238221

[9] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006. DOI: 10.1109/TIT.2006.874516

[10] G. R. Price, "Selection and covariance," *Nature*, vol. 227, pp. 520–521, 1970. DOI: 10.1038/227520a0

[11] G. Nguyen, *Bing turns 10: Why it's been more disruptive than you think*, https://searchengineland.com/bing-turns-10-why-its-been-more-disruptive-than-you-think-317510, Accessed March 2026, 2019.

[12] P. Aghion, U. Akcigit, and P. Howitt, "What do we learn from Schumpeterian growth theory?" In *Handbook of Economic Growth*, vol. 2, Elsevier, 2014, pp. 515–563. DOI: 10.1016/B978-0-444-53540-5.00001-X

[13] R. Ormándi, I. Hegedűs, and M. Jelasity, "Gossip learning with linear models on fully distributed data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 556–571, 2013, arXiv:1109.1396 (September 2011). DOI: 10.1002/cpe.2858

[14] B. Szörényi, R. Busa-Fekete, I. Hegedűs, R. Ormándi, M. Jelasity, and B. Kégl, "Gossip-based distributed stochastic bandit algorithms," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, ser. PMLR, vol. 28, 2013, pp. 19–27.

[15] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized collaborative learning of personalized models over networks," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. PMLR, vol. 54, 2017, pp. 509–517.

[16] A. Pacchiano et al., "Model selection in contextual stochastic bandit problems," in *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

[17] R. Chawla, A. Sankararaman, A. Ganesh, and S. Shakkottai, "The gossiping insert-eliminate algorithm for multi-agent bandits," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. PMLR, vol. 108, 2020, pp. 3471–3481.

[18] A. Lalitha and A. Goldsmith, "Bayesian algorithms for decentralized stochastic bandits," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 2, pp. 564–583, 2021. DOI: 10.1109/JSAIT.2021.3080661

[19] Z. Zhu, J. Zhu, J. Liu, and Y. Liu, "Federated bandit: A gossiping approach," *ACM SIGMETRICS Performance Evaluation Review*, vol. 49, no. 1, pp. 3–4, 2022. DOI: 10.1145/3543516.3453919

[20] B. Mueller, W. N. Street, S. Baek, Q. Lin, J. Yang, and Y. Huang, "FedPAE: Peer-adaptive ensemble learning for asynchronous and model-heterogeneous federated learning," in *Proceedings of the 2024 IEEE International Conference on Big Data (IEEE BigData)*, arXiv:2410.14075, 2024, pp. 7961–7970.

[21] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3–4, pp. 285–294, 1933. DOI: 10.1093/biomet/25.3-4.285

[22] Z. Hu, Y. Wang, and C. Chen, "Robust decentralized multi-armed bandits: From corruption-resilience to Byzantine-resilience," in *Proceedings of the 40th AAAI Conference on Artificial Intelligence (AAAI)*, arXiv:2511.10344, 2026.

[23] J. R. Douceur, "The Sybil attack," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, ser. Lecture Notes in Computer Science, vol. 2429, Springer, 2002, pp. 251–260. DOI: 10.1007/3-540-45748-8_24

[24] Y. Zhao et al., "Try before you buy: Solving multi-model complex tasks by model competitions," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025. DOI: 10.1109/ICASSP49660.2025.10890450

[25] I. Ong et al., "RouteLLM: Learning to route LLMs from preference data," in *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, arXiv:2406.18665, 2025.

[26] L. Chen, M. Zaharia, and J. Zou, "FrugalGPT: How to use large language models while reducing cost and improving performance," *Transactions on Machine Learning Research (TMLR)*, 2024, arXiv:2305.05176.

[27] W.-L. Chiang et al., "Chatbot arena: An open platform for evaluating LLMs by human preference," in *Proceedings of the 41st International Conference on Machine Learning (ICML)*, arXiv:2403.04132, 2024.

[28] F. Radlinski, R. Kleinberg, and T. Joachims, "Learning diverse rankings with multi-armed bandits," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008, pp. 784–791. DOI: 10.1145/1390156.1390255

[29] Y. Yue and T. Joachims, "Interactively optimizing information retrieval systems as a dueling bandits problem," in *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009, pp. 1201–1208. DOI: 10.1145/1553374.1553527

[30] A. Schuth, F. Sietsma, S. Whiteson, D. Lefortier, and M. de Rijke, "Multileaved comparisons for fast online evaluation," in *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (CIKM)*, 2014, pp. 71–80. DOI: 10.1145/2661829.2661952

[31] Tribler Team, Delft University of Technology, *Tribler: Privacy-enhanced BitTorrent client*, https://tribler.org/about.html, 2005.

[32] T. Qin and T.-Y. Liu, "Introducing LETOR 4.0 datasets," *arXiv preprint arXiv:1306.2597*, 2013.

[33] T. Qin and T.-Y. Liu, *MSLR-WEB10k dataset*, https://www.microsoft.com/en-us/research/project/mslr/, 2010.

[34] Istella SpA, *Istella LETOR dataset*, https://istella.ai/datasets/letor-dataset/, Accessed April 2026, 2016.

[35] M. Gregoriadis, J. Kang, and J. Pouwelse, "A large-scale web search dataset for federated online learning to rank," in *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM)*, 2025. DOI: 10.5281/zenodo.15678397

[36] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017, pp. 3146–3154.

[37] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, 2002. DOI: 10.1145/582415.582418

[38] H. Oosterhuis and M. de Rijke, "Differentiable unbiased online learning to rank," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*, 2018, pp. 1293–1302. DOI: 10.1145/3269206.3271686

[39] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785

[40] J. R. Holm, "The significance of structural transformation to productivity growth: How to account for levels in economic selection," *Journal of Evolutionary Economics*, vol. 24, no. 5, pp. 1009–1036, 2014.

[41] B. Nasrulin, G. Ishmaev, and J. Pouwelse, "Meritrank: Sybil tolerant reputation for merit-based tokenomics," in *2022 4th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, 2022, pp. 95–102. DOI: 10.1109/BRAINS55737.2022.9908685

[42] R. Chotkan, J. Decouchant, and J. Pouwelse, "Distributed attestation revocation in self-sovereign identity," *arXiv preprint arXiv:2208.05339*, 2022. DOI: 10.48550/arXiv.2208.05339

7 Appendix

Algorithm 2 Gossip-Augmented Decentralized MAB (Full)

Require: Arm set $\mathcal{K}$, identity i, queries-per-round Q, exploration constant c, minimum pulls $N_{\min}$, gossip interval T_g, observation TTL Δ_w, *algo* $\in \{\text{TS}, \text{UCB1}\}$, total pulls N
Operators (from Algorithm 1): prior s_0, aggregator γ (yields $\bar{c}_k$), index φ, update $\oplus$, eliminate ψ

Initialisation

$\mathcal{K}^{\text{exc}} \leftarrow \emptyset$; $L \leftarrow 0$; $q_{\text{round}} \leftarrow 0$ ▷ *local Lamport counter, round counter*
for each arm $k \in \mathcal{K}$ **do**
  $own_k \leftarrow s_0$ ▷ *prior s_0: $(\alpha{=}1, \beta{=}1, n{=}0)$ for TS; $(R{=}0, n{=}0)$ for UCB1*
  $\Pi_k \leftarrow \emptyset$ ▷ *$\Pi_k[s]$: latest observation of arm k from sender s*
  SEED(k) via libtorrent if local file exists
end for
$\Lambda \leftarrow \emptyset$ ▷ *$\Lambda[\langle s, k\rangle]$: highest Lamport stamp accepted from s for arm k*

Asynchronous handlers (run on incoming IPv8 messages)

on GOSSIPMSG$(s, k, n, R, \alpha, \beta, \ell, \text{magnet})$**:**
  if $k \notin \mathcal{K}$ **and** magnet $\neq \emptyset$**:**
    FETCHASYNC(k, magnet); **return** ▷ *discovery; arm enters $\mathcal{K}$ on FetchComplete*
  if $k \notin \mathcal{K}$**: return**
  if $\ell \leq \Lambda[\langle s, k\rangle]$**: return** ▷ *Lamport gating drops duplicates and out-of-order drops*
  $\Lambda[\langle s, k\rangle] \leftarrow \ell$
  $\Pi_k[s] \leftarrow (n, R, \alpha, \beta, \texttt{last\_received}{=}\text{NOW}())$ ▷ *overwrite, no merge*
  // triggered when libtorrent download finishes
on FETCHCOMPLETE(k)**:** SCORE(k); $\mathcal{K} \leftarrow \mathcal{K} \cup \{k\}$; $own_k \leftarrow s_0$; $\Pi_k \leftarrow \emptyset$ ▷ *prior s_0 for new arm*

Wall-clock gossip emitter (independent of query loop)

every T_g **seconds:**
  $k \leftarrow$ uniform random arm from $\mathcal{K} \setminus \mathcal{K}^{\text{exc}}$
  $j \leftarrow$ uniform random peer from IPv8's continuously-walked peer table
  if $k = \emptyset$ **or** $j = \emptyset$**:** skip this cycle
  $L \leftarrow L + 1$
  Send GOSSIPMSG$\big(i,\ k,\ own_k.n,\ own_k.R,\ own_k.\alpha,\ own_k.\beta,\ L,\ \text{magnet}_k\big)$ to j
  // TTL eviction: drop every $\Pi_k[s]$ with $\text{NOW}() - \Pi_k[s].\texttt{last\_received} > \Delta_w$

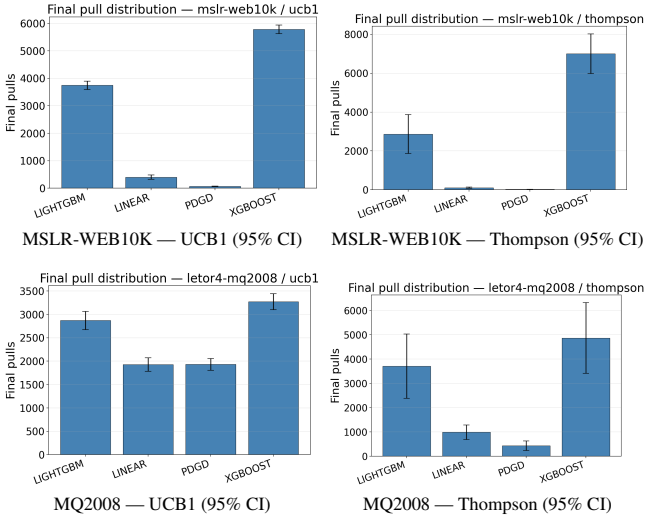
Main per-query loop

for query $q = 1, 2, \ldots$ **do**
  // Phase 1: aggregator γ — fold own + non-stale peer obs. into $\bar{c}_k$
  for each arm $k \in \mathcal{K}$ in deterministic order **do**
    $\mathcal{S}_k \leftarrow \{\, s \in \Pi_k : \text{NOW}() - \Pi_k[s].\texttt{last\_received} \leq \Delta_w \,\}$ ▷ *non-stale senders*
    if *algo* $=$ TS **then**
      $\hat{\alpha}_k \leftarrow 1 + (own_k.\alpha - 1) + \sum_{s \in \mathcal{S}_k} (\Pi_k[s].\alpha - 1)$
      $\hat{\beta}_k \leftarrow 1 + (own_k.\beta - 1) + \sum_{s \in \mathcal{S}_k} (\Pi_k[s].\beta - 1)$
      $\hat{n}_k \leftarrow own_k.n + \sum_{s \in \mathcal{S}_k} \Pi_k[s].n$ ▷ $(\hat{\alpha}_k, \hat{\beta}_k, \hat{n}_k) = \bar{c}_k$
    else if *algo* $=$ UCB1 **then**
      $\hat{R}_k \leftarrow own_k.R + \sum_{s \in \mathcal{S}_k} \Pi_k[s].R$; $\hat{n}_k \leftarrow own_k.n + \sum_{s \in \mathcal{S}_k} \Pi_k[s].n$ ▷ $(\hat{R}_k, \hat{n}_k) = \bar{c}_k$
    end if
  end for
  // Phase 2: index φ — select $k^* = \arg\max_k \varphi(\bar{c}_k)$ over non-excluded arms
  $\mathcal{A} \leftarrow \mathcal{K} \setminus \mathcal{K}^{\text{exc}}$
  if $\exists\, k \in \mathcal{A} : \hat{n}_k = 0$ **then**
    $k^* \leftarrow$ first such k in deterministic order ▷ *force-explore*
  else if *algo* $=$ TS **then**
    Sample $\theta_k \sim \text{Beta}(\hat{\alpha}_k, \hat{\beta}_k)$ for each $k \in \mathcal{A}$; $k^* \leftarrow \arg\max_{k \in \mathcal{A}} \theta_k$ ▷ *φ = posterior sample*
  else if *algo* $=$ UCB1 **then**
    $k^* \leftarrow \arg\max_{k \in \mathcal{A}} \left[\dfrac{\hat{R}_k}{\hat{n}_k} + c\sqrt{\dfrac{\ln(N+1)}{\hat{n}_k + 1}} \right]$; ties broken by RNG ▷ *$\varphi$ = UCB index*

end if
// Phase 3: update $\oplus$ — $own_{k^*} \leftarrow own_{k^*} \oplus r$
ranking $\leftarrow \textsc{Model}_{k^*}(q, D)$; $r \leftarrow \text{NDCG@10}(\textit{ranking})$
$L \leftarrow L + 1$; $own_{k^*}.\texttt{lamport} \leftarrow L$; $own_{k^*}.n \mathrel{+}= 1$
if *algo* $=$ TS **then**
  $own_{k^*}.\alpha \mathrel{+}= r$; $own_{k^*}.\beta \mathrel{+}= (1 - r)$ ▷ $\oplus$ *for* TS
else if *algo* $=$ UCB1 **then**
  $own_{k^*}.R \mathrel{+}= r$ ▷ $\oplus$ *for* UCB1
end if
$q_{\text{round}} \leftarrow q_{\text{round}} + 1$
// Round boundary: local elimination only (gossip is independent, see emitter)
if $q_{\text{round}} < Q$ **then continue**
end if
$q_{\text{round}} \leftarrow 0$
// Phase 4: eliminate ψ — $\psi(\bar{c}_k, \{\bar{c}_{k'}\})$ *via LCB/UCB domination (purely local; no broadcast)*
$\mathcal{E} \leftarrow \{\, k \in \mathcal{K} \setminus \mathcal{K}^{\text{exc}} : \hat{n}_k \geq N_{\min} \,\}$
if $|\mathcal{E}| > 1$ **then**
  for each arm $k \in \mathcal{E}$ **do**
    $\hat{\mu}_k \leftarrow \hat{R}_k/\hat{n}_k$ (UCB1) **or** $\hat{\alpha}_k/(\hat{\alpha}_k + \hat{\beta}_k)$ (TS) ▷ *point estimate from* $\bar{c}_k$
    $w_k \leftarrow \sqrt{\ln(N+1)/(2\hat{n}_k)}$
    $\text{LB}_k \leftarrow \max(0, \hat{\mu}_k - w_k)$; $\text{UB}_k \leftarrow \min(1, \hat{\mu}_k + w_k)$
  end for
  $k_{\text{best}} \leftarrow \arg\max_{k \in \mathcal{E}} \text{LB}_k$
  for each arm $k \in \mathcal{E},\ k \neq k_{\text{best}}$ **do**
    if $\text{UB}_k < \text{LB}_{k_{\text{best}}}$ **then** ▷ ψ *fires:* k *confidently dominated*
      $\mathcal{K}^{\text{exc}} \leftarrow \mathcal{K}^{\text{exc}} \cup \{k\}$ ▷ *remove from choice set only; aggregate stats persist*
    end if
  end for
end if
end for

Additional Experimental Setup Details

All experiment configurations are repeated with the same fixed set of random seeds: $\{42, 1337, 2718, 3141, 161803, 271828, 577215, 14142, 86420, 99999\}$. Plotted curves and tables aggregate these ten runs, and shaded bands denote 95% confidence intervals.



MSLR-WEB10K — UCB1 (95% CI) MSLR-WEB10K — Thompson (95% CI)

MQ2008 — UCB1 (95% CI) MQ2008 — Thompson (95% CI)

Figure 7: Final arm-pull distributions for representative configurations: the largest dataset (MSLR-WEB10K, top) and the smallest (MQ2008, bottom), for both UCB1 (left) and Thompson Sampling (right). Pulls concentrate on the highest-mean model, with Thompson exploiting more aggressively.