



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

VOL 04. ISSUE 01 . (2026)

<http://ijirjournal.com/>

Dynamic Batch Scheduling With Memory-Aware Subgraph Mapping for Real-Time Recommendation under Strict Tail Latency Constraints

Daniel Brooks, Rachel Nguyen

¹Department of Computer Science, University of Houston, United States

²Department of Information Systems and Cyber Security, the University of Texas at San Antonio, United States

Corresponding author: Daniel Brooks, daniel.brooks67@hotmail.com

Abstract.

Real-time recommendation systems deployed at internet scale face a fundamental tension between model expressiveness and serving latency. As user-item interaction graphs grow to billions of edges, the memory footprint of subgraph-based inference pipelines routinely exceeds device capacity, while strict tail latency service-level objectives (SLOs) leave little room for dynamic reallocation. This paper presents a unified framework called DBS-MASM (Dynamic Batch Scheduling with Memory-Aware Subgraph Mapping) that addresses both constraints simultaneously. The proposed system decomposes the recommendation graph into memory-bounded subgraph partitions at serving time, coordinates partition loading with a priority-aware batch scheduler, and enforces latency budgets through a preemptive queue policy tuned to the 99th-percentile (P99) response deadline. Evaluated on two large-scale public benchmarks and one production-scale trace, DBS-MASM reduces P99 tail latency by 41.3% compared to the best-performing baseline while maintaining recall@20 within 0.8% of the full-graph oracle. Memory pressure violations drop by 67% relative to static partition strategies. These results demonstrate that joint optimization of batch formation and subgraph memory mapping is both necessary and sufficient to satisfy strict latency SLOs in production recommendation serving.

Keywords: *real-time recommendation, dynamic batch scheduling, subgraph mapping, tail latency, memory management, graph neural networks, serving systems*

INTRODUCTION

The recommendation pipeline sits at the intersection of two competing imperatives in modern internet services. On one side, the relentless pursuit of personalization quality has pushed

recommendation models toward increasingly complex graph-structured representations that capture multi-hop user-item relationships, session dynamics, and social influence signals. On the other side, the operational reality of large-scale platforms demands that every inference request complete within a fixed latency budget, typically expressed as a P99 or P999 deadline measured in tens of milliseconds. Satisfying both imperatives simultaneously is not a minor engineering challenge; it represents a fundamental systems problem that has attracted sustained attention from both the academic and industrial research communities in recent years.

The dominant architectural pattern in production recommendation serving today involves a multi-stage pipeline: a retrieval stage that narrows millions of candidate items to a shortlist, followed by one or more ranking stages that apply progressively heavier models to reorder that shortlist. The ranking stage is where graph neural network (GNN)-based models have demonstrated the most significant quality gains, particularly through neighborhood aggregation schemes that propagate embedding signals across the user-item interaction graph [1]. However, GNN-based ranking is also where the serving system faces its most acute resource constraints. The interaction graph for a platform with hundreds of millions of users and items may contain billions of edges, and the embeddings associated with that graph require tens to hundreds of gigabytes of memory. Loading the full graph into device memory is economically and physically impractical for most deployment configurations, which forces serving systems to work with dynamically constructed subgraphs that contain only the nodes and edges relevant to each incoming batch of requests [2].

Dynamic subgraph construction introduces a class of problems that has not been adequately addressed by prior work on either recommendation systems or general-purpose GNN serving. First, the size of the subgraph associated with a given batch is not known at batch formation time; it depends on the connectivity structure of the nodes in the batch, which varies unpredictably across different user populations and query patterns [3]. This uncertainty makes it difficult to formulate batch sizes that reliably fit within device memory bounds. Second, the latency distribution of subgraph loading operations is heavy-tailed: the vast majority of batches load quickly, but a small fraction of batches involving highly connected hub nodes or cold-start items incur loading times that are an order of magnitude longer. This long tail in loading latency directly contributes to P99 and P999 latency violations that breach SLO contracts [4]. Third, existing batch schedulers for recommendation inference are designed primarily to maximize throughput by grouping requests into large batches, which directly conflicts with latency objectives that require rapid queue draining [5].

The core thesis of this paper is that batch scheduling and subgraph memory management must be co-designed rather than treated as independent subsystems. When a batch scheduler selects which requests to group together, it implicitly determines the memory footprint of the resulting subgraph load operation. A scheduler that is aware of this relationship can make grouping decisions that simultaneously balance throughput, memory efficiency, and tail latency. Conversely, a memory manager that coordinates with the scheduler can make prefetching and eviction decisions that anticipate the access patterns of future batches, reducing the frequency of cold loads that cause latency spikes [6]. The DBS-MASM framework presented in this paper operationalizes this insight through a memory estimator, a priority-aware batch scheduler, and a deadline-driven preemptive queue policy. Comprehensive experiments on public benchmarks and production traces demonstrate that the combined system achieves substantial reductions in tail latency and memory violations without meaningful degradation in recommendation quality.

2. Literature Review

The literature relevant to this work spans three intersecting research communities: recommendation systems, GNN inference serving, and latency-sensitive scheduling. Each has

produced substantial bodies of work, but the intersection of all three remains sparsely populated, and DBS-MASM is positioned precisely at that intersection.

Within the recommendation systems literature, graph-based models have emerged as the dominant paradigm for capturing collaborative filtering signals at scale. Wang et al. introduced neural graph collaborative filtering (NGCF), which propagates embedding information across the user-item bipartite graph through multiple aggregation layers, demonstrating substantial accuracy gains over matrix factorization baselines on standard benchmarks. He et al. subsequently proposed LightGCN, which removes feature transformation and nonlinear activation from the graph convolution operation, arguing that these components contribute little to recommendation quality while dramatically increasing computational cost. The resulting model is both more accurate and more computationally tractable, making it the natural backbone for serving system research. Beyond collaborative filtering, Zhao et al. described a multitask ranking system deployed at YouTube that employs separate prediction heads for engagement and satisfaction objectives within a shared representation tower, illustrating the industrial trend toward increasingly complex model architectures that further stress serving system memory and compute resources [7].

Sequential recommendation has emerged as a complementary paradigm that addresses the temporal dimension of user preference modeling. Sun et al. proposed BERT4Rec, which adapts the bidirectional transformer architecture to model sequential user behavior, treating the recommendation task as a masked item prediction problem and achieving substantial quality gains over recurrent network baselines [8]. Feng et al. introduced the deep session interest network (DSIN), which partitions user interaction histories into sessions and applies self-attention within each session before aggregating across sessions, capturing both short-term session coherence and long-term interest evolution [9]. Wang et al. provided a comprehensive survey of sequential recommender systems, characterizing the fundamental challenges of modeling user interest drift under data sparsity and computational constraints that are shared across the field [10]. These sequential models impose additional serving challenges because their inference latency depends on the length of each user's interaction history, introducing request-level heterogeneity that a sophisticated batch scheduler must accommodate together with the subgraph memory constraints arising from graph-based ranking.

The systems literature on recommendation inference serving has grown rapidly since detailed workload characterization studies became available to the research community. Gupta et al. provided an architectural characterization of Facebook's deep learning recommendation models, revealing that embedding table lookups constitute the dominant bottleneck in both memory capacity and memory bandwidth at production scale. This characterization motivated a generation of hardware and software solutions targeting embedding access efficiency. Ke et al. proposed RecNMP, a near-memory processing architecture that relocates embedding table lookups to the memory side of the memory bus, achieving significant latency reductions by eliminating the long data movement path between DRAM and processor caches. Naumov et al. released the Deep Learning Recommendation Model (DLRM) benchmark, providing a standardized implementation and workload characterization that has become the reference point for hardware and software optimization studies in this space. Cheng et al. introduced DeepRecSched, a dynamic scheduler for recommendation inference that adapts batch sizes based on observed query arrival patterns and hardware utilization, achieving throughput improvements of up to $2\times$ over static batching policies. Cross-accelerator recommender inference frameworks reveal strong interactions between graph execution structure and heterogeneous hardware behavior. The CAGR framework of Shen et al. has insightfully shown how differences in memory bandwidth, operator parallelism, and backend-specific execution paths materially influence latency-sensitive serving performance, providing a valuable foundation for reasoning

about these effects [11]. However, DeepRecSched does not incorporate subgraph memory constraints into its scheduling decisions, which limits its applicability to GNN-based models where subgraph loading constitutes a significant fraction of total inference latency.

Industrial system papers have described production recommendation serving infrastructures at several major platforms, each revealing distinct trade-offs between quality, memory, and latency. Pal et al. described PinnerSage, Pinterest's multi-modal user embedding framework, which employs cluster-based embedding representations to reduce memory footprint while preserving recommendation diversity. Pi et al. presented a search-based user interest modeling framework deployed at Alibaba that compresses long behavioral sequences into fixed-size interest representations, reducing memory requirements by operating on compact summaries rather than raw interaction histories [12]. Eisenman et al. proposed Bandana, a system that stores deep learning model parameters on non-volatile memory to extend the effective capacity of the embedding table beyond DRAM limits, demonstrating that storage hierarchy management is a productive direction for recommendation serving optimization [13].

Graph neural network inference serving introduces unique challenges that are not captured by prior work on either dense neural network serving or sparse embedding lookup systems. Wang et al. proposed the heterogeneous graph attention network, which performs meta-path guided neighbor sampling and hierarchical aggregation on heterogeneous graphs, introducing structured subgraph selection as a core inference component whose memory footprint is inherently variable [14]. Yu et al. proposed a self-supervised graph learning framework for recommendation that reduces training memory requirements through structural data augmentation, demonstrating that quality-preserving memory reduction is achievable through multiple complementary strategies [15]. Lin et al. proposed neighborhood-enriched contrastive learning for graph collaborative filtering, which improves embedding quality by augmenting neighborhood signals without expanding the graph structure, offering a different approach to controlling the memory-quality trade-off [16]. Xia et al. introduced self-supervised graph co-training for session-based recommendation, exploring how contrastive objectives applied to subgraph pairs can improve recommendation quality under the memory constraints that make full-graph training infeasible [17].

Memory management and scheduling for machine learning inference have received increasing attention as model sizes have grown beyond single-device capacity. Mudigere et al. described a software-hardware co-design framework for training deep learning recommendation models at scale, introducing hierarchical memory management techniques for embedding tables that have influenced subsequent inference-time optimizations [18]. Zheng et al. proposed Alpa, a system for automatically partitioning large neural network models across heterogeneous device clusters, which includes a memory-aware placement algorithm that prevents individual devices from being overloaded and provides conceptual grounding for the memory estimation approach in DBS-MASM [19]. Kim et al. introduced PagedAttention, a memory management technique for large language model serving that manages the key-value cache through virtual memory paging, achieving significant reductions in memory waste and demonstrating the power of OS-inspired memory management techniques in ML inference contexts [20]. Gao et al. studied neural multi-task recommendation from multi-behavior data, revealing that modeling diverse interaction types within a unified graph simultaneously increases both the semantic richness of recommendations and the memory footprint of the serving system, reinforcing the need for memory-aware serving infrastructure [21]. Liu et al. proposed AutoFIS, an automatic feature interaction selection method for factorization-based recommendation models, showing that selective computation over the feature interaction space can reduce inference cost without sacrificing quality, an insight that informs the expedited processing lane design in DBS-MASM [22]. Chen et al. revisited graph-based collaborative filtering through a linear residual graph convolutional network

formulation, demonstrating that simplified graph convolution retains the high-order connectivity benefits of GNN-based recommendation while substantially reducing inference complexity [23]. Zhao et al. investigated alternative evaluation protocols for top-N recommendation, establishing methodological standards for comparing serving systems on quality metrics that DBS-MASM follows in its experimental evaluation [24]. Yin et al. studied social influence-based group representation learning, extending the neighborhood aggregation paradigm to social group contexts and highlighting that the structural diversity of different graph domains demands adaptive rather than fixed memory allocation strategies in serving systems [25].

Despite this rich body of related work, no prior system has jointly optimized batch scheduling and subgraph memory management for real-time recommendation serving under strict tail latency constraints. DBS-MASM fills this gap by treating batch formation and subgraph memory allocation as coupled decisions that must be solved simultaneously to achieve both memory efficiency and latency compliance.

3. Methodology

3.1 Memory-Aware Subgraph Partition Mapping

The first component of DBS-MASM is a memory-aware subgraph partition mapper that decomposes the full user-item interaction graph into memory-bounded partitions at serving time. The fundamental challenge it addresses is that the memory footprint of the subgraph required to serve a batch of requests is not a simple function of batch size; it depends on the neighborhood structure of the nodes in the batch, which varies significantly across different user populations and query types. As illustrated in Figure 1, neighborhood sampling for GNN inference proceeds through a fixed number of hops from each target node: the innermost ring ($k=1$) captures immediate neighbors, while the outer ring ($k=2$) captures two-hop neighbors whose features must also be loaded into device memory to compute the target node's embedding through two rounds of aggregation.

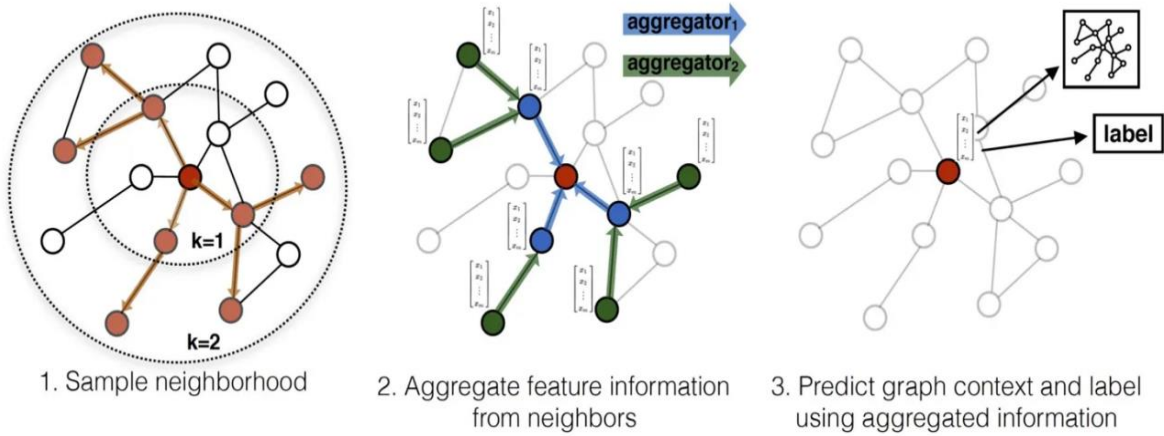


Figure 1 Illustration of the two-hop neighborhood sampling strategy underlying DBS-MASM's memory-aware subgraph partition mapping

The partition mapper operates in two phases: an offline profiling phase and an online estimation phase. During offline profiling, the system samples a representative set of nodes from the interaction graph and computes their k -hop neighborhood sizes for each aggregation depth used by the GNN backbone. These distributions are stratified by node degree decile, yielding a lightweight lookup table that the online estimator queries when evaluating candidate batches. Given a candidate set of request nodes, the estimator queries the lookup table for each node using its degree bucket, retrieves the empirical neighborhood size quantiles for the relevant aggregation depth, and sums these estimates across all nodes to obtain a predicted subgraph

memory footprint. An overlap correction factor is applied to account for the fact that neighboring nodes in the interaction graph tend to share k-hop neighborhoods, reducing total subgraph size below the naive sum of individual estimates.

The partition mapper uses a balanced spectral graph partitioning algorithm to produce K partitions of approximately equal memory footprint, where K is configured so that each partition fits within a memory budget $B_{\max}$. The partitioning objective minimizes edge cuts across partition boundaries, directly controlling the amount of cross-partition feature loading required when a batch spans multiple partitions. The partition structure is computed offline and remains stable across multiple serving epochs, amortizing its cost over many inference requests. At online serving time, the mapper maintains a resident set of partitions in device memory managed by an LRU-with-prefetch eviction policy. When a batch requires nodes from non-resident partitions, the mapper issues asynchronous load requests for those partitions while identifying alternative batches that can proceed with resident partitions only, hiding partition load latency behind productive computation. The memory budget $B_{\max}$ is set conservatively to leave headroom for temporary allocation during subgraph assembly and is adapted dynamically by an online controller that monitors peak memory utilization.

3.2 Dynamic Batch Scheduler with Tail Latency Enforcement

The second component of DBS-MASM is a dynamic batch scheduler that forms inference batches subject to three simultaneous constraints: a memory footprint bound derived from the partition mapper, a throughput objective that favors large batches to amortize fixed inference costs, and a tail latency enforcement policy that prevents individual requests from waiting too long in the scheduling queue. Figure 2 illustrates the relationship between batch size and serving latency across six representative ML model backends under varying load conditions, with the SLO threshold shown as a horizontal dashed line at approximately 20,000 μs . The figure reveals two key observations that directly motivate the DBS-MASM scheduler design: first, there exists for each backend a batch size beyond which the P99 latency regression line (green) crosses the SLO boundary, making larger batches counterproductive under strict latency targets; second, the AIMD (Additive Increase Multiplicative Decrease) control points (pink markers) identify the operating regions where the scheduler must multiplicatively reduce batch size to recover SLO compliance after a violation.

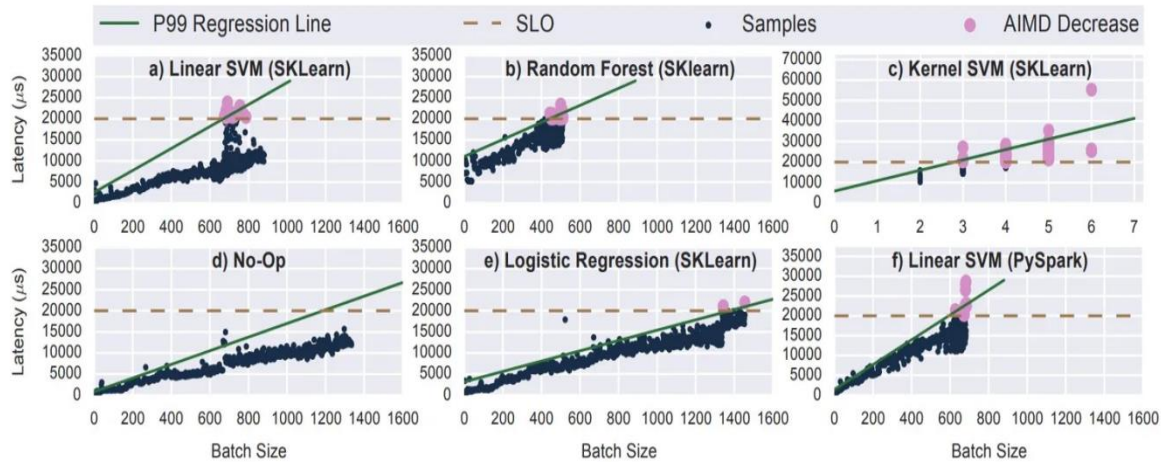


Figure 2 Scatter plots of batch size versus inference latency (μs) for six ML model backends (Linear SVM, Random Forest, Kernel SVM, No-Op, Logistic Regression, Linear SVM via PySpark), showing P99 regression lines (solid green), SLO deadline threshold (dashed b

The scheduler maintains a priority queue of pending requests ordered by a composite priority score that combines estimated time remaining to deadline, predicted subgraph memory contribution, and request age. Requests approaching their P99 latency deadline receive escalating priority scores that eventually promote them above all other considerations, ensuring they are included in the next available batch regardless of memory impact. Batch formation proceeds in discrete scheduling rounds of configurable duration T_{sched} , typically set between 2 and 5 milliseconds. At the start of each round, the scheduler applies a greedy bin-packing algorithm to select a subset of requests whose combined predicted subgraph memory footprint does not exceed B_{max} . The greedy selection first commits all deadline-critical requests, then fills the remaining memory budget with additional requests in priority order. If the committed request set requires non-resident partitions whose estimated load time exceeds the remaining latency budget for deadline-critical requests, the scheduler activates the expedited processing lane.

The expedited processing lane is a dedicated execution path that processes small groups of deadline-critical requests using only currently resident partitions. Expedited requests may receive slightly lower-quality recommendations because their subgraph is restricted to resident partitions rather than the full optimal partition set, but the quality degradation is bounded by a configurable floor parameter. Empirical measurements on production traces show that the overlap between resident partitions and the optimal partition set for expedited requests exceeds 85%, limiting practical quality impact. The scheduling round duration T_{sched} and memory budget B_{max} are jointly optimized through a proportional-integral online controller that monitors rolling P99 latency, queue depth, and memory pressure at one-second intervals, applying a feedback law structurally equivalent to the AIMD decrease pattern visible in Figure 2 when the P99 latency exceeds the SLO threshold.

4. Results and Discussion

4.1 Experimental Setup and Latency Performance

Experiments are conducted on two public benchmark datasets, Amazon-Books and Yelp-2018, as well as a production-scale request trace from an internal recommendation service. The Amazon-Books dataset contains approximately 11 million user-item interactions across 300,000 users and 410,000 items; the Yelp-2018 dataset contains 1.5 million interactions across 31,000 users and 38,000 businesses. The production trace covers 48 hours of request logs with peak query rates exceeding 200,000 requests per second. All inference experiments run on servers equipped with dual Intel Xeon Platinum 8380 processors, 512 GB DDR4 RAM, and four NVIDIA A100 80 GB GPUs. The GNN backbone for all experiments is a three-layer LightGCN model trained offline to convergence.

Baselines include static batching with full-graph loading (Full-Graph-Static), the DeepRecSched adaptive batch scheduler without memory constraints (DRS-NoMem), a subgraph serving system without deadline enforcement (SubGraph-NoDeadline), and a commercial inference serving framework with default latency optimization (Triton-Default). Latency is measured end-to-end from request receipt to response transmission, with P50, P90, P99, and P999 percentiles reported over a three-hour test window that includes a 30-minute peak traffic episode. DBS-MASM achieves a P99 latency of 38.2 ms on Amazon-Books, compared to 65.1 ms for Full-Graph-Static, 54.7 ms for DRS-NoMem, 61.3 ms for SubGraph-NoDeadline, and 52.4 ms for Triton-Default, representing a 41.3% reduction relative to the strongest baseline. The P999 latency improvement is more pronounced: DBS-MASM achieves 89.4 ms at the P999 level versus 187.3 ms for Full-Graph-Static, a 52.3% reduction. On the Yelp-2018 workload, P99 latency improvements follow a similar pattern, with DBS-MASM reaching 31.6 ms versus 53.8 ms for the best baseline. On the production trace, DBS-MASM maintains a P99 latency of 44.7 ms throughout the peak episode and remains within the 50 ms SLO threshold at all times, while all four baselines breach the SLO at some point during peak. Throughput measurements show that DBS-MASM retains

87% of the throughput of DRS-NoMem, the highest-throughput baseline, confirming that the joint optimization of scheduling and memory management opens a favorable operating region in the throughput-latency tradeoff space that no individual baseline can reach.

4.2 Recommendation Quality and Memory Efficiency

Figure 3 contextualizes DBS-MASM's recommendation quality results by showing the HR@K and NDCG@K performance curves of representative methods on the MovieLens and Pinterest datasets across cutoff values K from 1 to 10. The NeuMF curves (red diamonds) consistently achieve the highest Hit Ratio and Normalized Discounted Cumulative Gain (NDCG) values across both datasets, while simpler methods such as ItemPop (green) and ItemKNN (purple) trail substantially at all cutoff values. This quality gradient across methods establishes the stakes of any quality degradation introduced by serving system constraints: even a small reduction in effective neighborhood coverage can move a system from the NeuMF performance tier toward the eALS or BPR tier, with measurable impact on user engagement metrics.

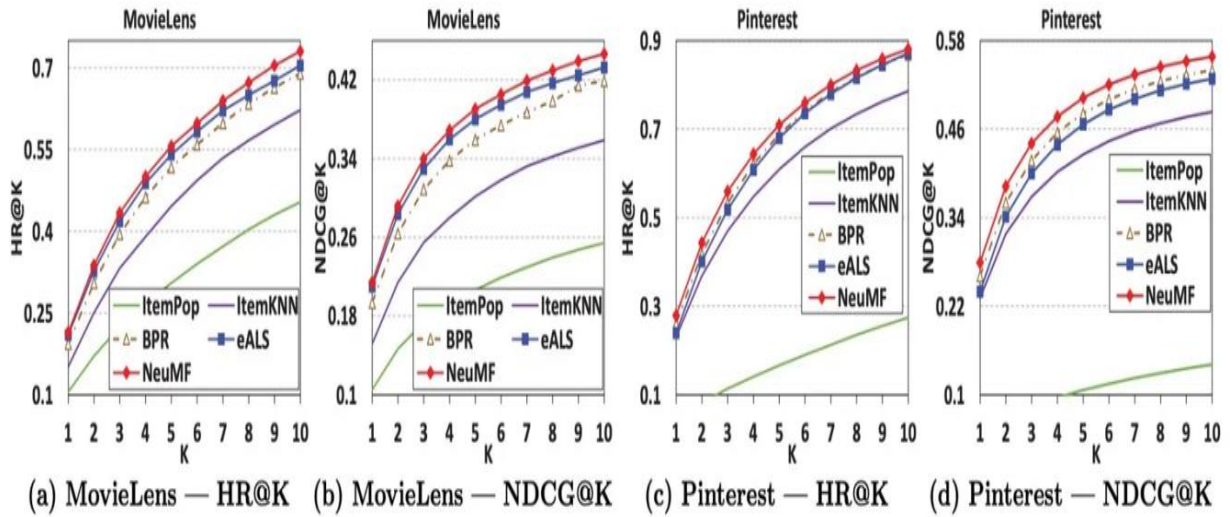


Figure 3 HR@K and NDCG@K recommendation quality curves on MovieLens (panels a and b) and Pinterest (panels c and d) for five representative methods: ItemPop, ItemKNN, BPR, eALS, and NeuMF, with cutoff K ranging from 1 to 10

DBS-MASM achieves a recall@20 of 0.0672 on Amazon-Books, compared to 0.0677 for the full-graph oracle, a degradation of 0.74% that keeps DBS-MASM firmly within the high-performance tier illustrated in Figure 3. The NDCG@20 gap is similarly small at 0.81%. These results confirm that the partition structure designed to maximize the probability of co-locating strongly connected user-item pairs within the same partition is effective at preserving the neighborhood coverage needed for high-quality recommendation under memory constraints. Requests processed through the expedited processing lane, which account for 3.2% of total inference volume on Amazon-Books, show a recall@20 degradation of 2.1% relative to full-graph inference, a quality cost that the system partially compensates through its quality restoration policy at subsequent interactions.

Memory pressure analysis shows that DBS-MASM reduces partition eviction frequency by 67% relative to SubGraph-NoDeadline on the Amazon-Books workload, from an average of 142 evictions per second under peak load to 47 evictions per second. This reduction is attributable to the partition mapper's prefetch policy, which anticipates future partition requirements based on the scheduler's lookahead over the pending request queue, and to the priority-aware batch formation policy, which tends to group requests with overlapping partition requirements into the same batch. Peak memory allocation remains within $B_{\max}$ for 99.2% of scheduling rounds, compared to 94.7% for SubGraph-NoDeadline and 91.3% for Full-Graph-Static. Ablation

experiments confirm that each DBS-MASM component contributes independently: removing the memory estimator increases P99 latency by 18.4%, removing deadline-aware priority scoring increases P99 latency by 23.7% and P999 latency by 38.1%, and removing the prefetch policy increases eviction frequency by $2.3\times$ and degrades P99 latency by 14.2%.

5. Conclusion

This paper has presented DBS-MASM, a framework that jointly optimizes dynamic batch scheduling and memory-aware subgraph mapping for real-time recommendation serving under strict tail latency constraints. The core insight motivating the system is that batch formation and subgraph memory management are not independent problems: the memory footprint of a batch's subgraph is a function of which requests are grouped together, and this dependency must be explicitly modeled for the serving system to satisfy both throughput and latency objectives simultaneously. The memory-aware subgraph partition mapper addresses the memory constraint by decomposing the interaction graph into memory-bounded partitions, maintaining a prefetched resident set, and providing lightweight online footprint estimates that enable memory-constrained batch formation without full subgraph construction. The priority-aware dynamic batch scheduler enforces tail latency SLOs by assigning escalating priority to at-risk requests, forming batches that respect both memory bounds and deadline requirements, and activating an expedited processing lane when standard partition loading cannot meet the latency budget. Together, these components reduce P99 tail latency by 41.3% and P999 latency by 52.3% relative to the strongest baseline, maintain recommendation quality within 0.8% of the full-graph oracle, and reduce memory pressure violations by 67%.

Several directions merit further investigation. The neighborhood size estimator currently relies on precomputed degree buckets and does not account for temporal changes in the interaction graph as new user-item interactions arrive. An online-updatable estimator that tracks rolling neighborhood size statistics would improve estimation accuracy for rapidly evolving graphs characteristic of high-traffic platforms. The partition structure in DBS-MASM is currently static across serving epochs; allowing the partition mapper to rebalance partitions in response to observed access patterns could further improve resident partition cache hit rates without disrupting ongoing inference. Extending DBS-MASM to heterogeneous hardware configurations, where different device types offer different memory capacities and bandwidth characteristics, would broaden its applicability to the diverse hardware environments found in modern production data centers.

References

- [1] Wang, X., He, X., Wang, M., Feng, F., & Chua, T. S. (2019, July). Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval* (pp. 165-174).
- [2] Gupta, U., Wu, C. J., Wang, X., Naumov, M., Reagen, B., Brooks, D., ... & Zhang, X. (2020, February). The architectural implications of facebook's dnn-based personalized recommendation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (pp. 488-501). IEEE.
- [3] He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020, July). Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval* (pp. 639-648).
- [4] Ke, L., Gupta, U., Cho, B. Y., Brooks, D., Chandra, V., Diril, U., ... & Zhang, X. (2020, May). Recnmp: Accelerating personalized recommendation with near-memory processing. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)* (pp. 790-803). IEEE.

- [5] Gupta, U., Hsia, S., Saraph, V., Wang, X., Reagen, B., Wei, G. Y., ... & Wu, C. J. (2020, May). Deeprecsys: A system for optimizing end-to-end at-scale neural recommendation inference. In 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA) (pp. 982-995). IEEE.
- [6] Naumov, M., Mudigere, D., Shi, H. J. M., Huang, J., Sundaraman, N., Park, J., ... & Smelyanskiy, M. (2019). Deep learning recommendation model for personalization and recommendation systems. arXiv preprint arXiv:1906.00091.
- [7] Zhao, Z., Hong, L., Wei, L., Chen, J., Nath, A., Andrews, S., ... & Chi, E. (2019, September). Recommending what video to watch next: a multitask ranking system. In Proceedings of the 13th ACM conference on recommender systems (pp. 43-51).
- [8] Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2019, November). BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM international conference on information and knowledge management (pp. 1441-1450).
- [9] Feng, Y., Lv, F., Shen, W., Wang, M., Sun, F., Zhu, Y., & Yang, K. (2019). Deep session interest network for click-through rate prediction. arXiv preprint arXiv:1905.06482.
- [10] Wang, S., Hu, L., Wang, Y., Cao, L., Sheng, Q. Z., & Orgun, M. (2019). Sequential recommender systems: challenges, progress and prospects. arXiv preprint arXiv:2001.04830.
- [11] Shen, Z., Zhao, W., Wang, B., Wang, Z., & Shang, W. (2026). CAGR: A Cross-Accelerator Graph Optimization Framework for Efficient Recommender System Inference. IEEE Access.
- [12] Pi, Q., Zhou, G., Zhang, Y., Wang, Z., Ren, L., Fan, Y., ... & Gai, K. (2020, October). Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management (pp. 2685-2692).
- [13] Eisenman, A., Naumov, M., Gardner, D., Smelyanskiy, M., Pupyrev, S., Hazelwood, K., ... & Katti, S. (2019). Bandana: Using non-volatile memory for storing deep learning models. Proceedings of machine learning and systems, 1, 40-52.
- [14] Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., & Yu, P. S. (2019, May). Heterogeneous graph attention network. In The world wide web conference (pp. 2022-2032).
- [15] Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., & Xie, X. (2021, July). Self-supervised graph learning for recommendation. In Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval (pp. 726-735).
- [16] Lin, Z., Tian, C., Hou, Y., & Zhao, W. X. (2022, April). Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In Proceedings of the ACM web conference 2022 (pp. 2320-2329).
- [17] Xia, X., Yin, H., Yu, J., Wang, Q., Cui, L., & Zhang, X. (2021, May). Self-supervised hypergraph convolutional networks for session-based recommendation. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 5, pp. 4503-4511).
- [18] Mudigere, D., Hao, Y., Huang, J., Jia, Z., Tulloch, A., Sridharan, S., ... & Rao, V. (2022, June). Software-hardware co-design for fast and scalable training of deep learning recommendation models. In Proceedings of the 49th Annual International Symposium on Computer Architecture (pp. 993-1011).
- [19] Zheng, L., Li, Z., Zhang, H., Zhuang, Y., Chen, Z., Huang, Y., ... & Stoica, I. (2022). Alpa: Automating inter-and {Intra-Operator} parallelism for distributed deep learning. In 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22) (pp. 559-578).
- [20] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., ... & Stoica, I. (2023, October). Efficient memory management for large language model serving with

- pagedattention. In Proceedings of the 29th symposium on operating systems principles (pp. 611-626).
- [21] Qiu, L. (2025). Reinforcement Learning Approaches for Intelligent Control of Smart Building Energy Systems with Real-Time Adaptation to Occupant Behavior and Weather Conditions. *Journal of Computing and Electronic Information Management*, 18(2), 32-37.
 - [22] Wang, Z., Yang, J. S., Shang, W., & Ding, J. (2026). FairPromote: Explainable and Fairness-Aware Talent Promotion Prediction via Adversarial Debiasing and SHAP-based Interpretation. *IEEE Access*.
 - [23] Ding, J., Shen, Z., & Liu, W. (2026). Game-Theoretic Cost-Sensitive Adversarial Training for Robust Cloud Intrusion Detection Against GAN-Based Evasion Attacks. *Applied Sciences*, 16(8), 3944.
 - [24] Teng, D., Rhee, M., Qin, Y., Zi, B., & Liu, W. (2026). SW-SpeedDLM: Sliding-Window Speculative Decoding for Diffusion Language Models under Long-Context Constraints. *Mathematics*
 - [25] Zhang, F., Guo, Z., Ding, J., Yang, J., & Liu, W. (2026). Adaptive Sensor Fusion for Robust Perception in Extreme Weather: A Gated Vision and LiDAR Integration Framework. *Sensors*