International Journal of Computer Science and Information Technology
ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 8, Number 6, Year 2026
DOI: https://doi.org/10.62051/ijcsit.v8n6.02
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Research on Deep Reinforcement Learning-Based Multi-Table Join Order Selection

Shenli Qi *

School of Artificial Intelligence, Guangzhou Maritime University, Guangzhou, Guangdong, 510725, China

*Corresponding Author: shenlichic@gmail.com

ABSTRACT

This paper presents a comprehensive survey of join order selection (JOS) in database query optimization, with a particular focus on deep reinforcement learning (DRL)-based approaches. By synthesizing representative systems such as Google Balsa and Microsoft Adaptive Query Processing, we characterize current trends and open challenges in reinforcement learning-driven query optimization. We formulate the multi-way join ordering problem as a Markov Decision Process (MDP) and systematically compare key design dimensions, including state representation (e.g., graph neural networks versus vectorized statistical features), action space design (e.g., join pair selection versus join subtree expansion), and reward function design (e.g., sparse versus intermediate rewards), analyzing their impacts on optimization outcomes. For multi-objective optimization, we survey strategies such as Pareto optimization and scalarization, highlighting their effectiveness in balancing query latency reduction with resource consumption and execution stability. To enhance model generalization, we examine the roles of curriculum learning and meta-learning in improving sample efficiency and robustness across heterogeneous workloads. In addition, we categorize hybrid architectures that integrate DRL with traditional cost-based optimizers (CBOs), including designs where DRL functions as a query rewriter, a search-space guide for plan enumeration, or a post-optimization validator. Finally, extensive empirical evaluations on standard benchmarks such as TPC-H and the Join Order Benchmark (JOB) demonstrate that DRL-based methods can outperform classical optimizers, particularly for complex queries with large search spaces.

KEYWORDS

Deep Reinforcement Learning (DRL); Multi-way Join Optimization; Markov Decision Process (MDP); Meta-learning; Hybrid Architectures

1. INTRODUCTION

The rapid advancement of the digital economy has led to exponential growth in global data volume. According to projections by International Data Corporation, the total global data volume is expected to reach 175 zettabytes by 2025. In the era of big data, multi-way join queries represent one of the most computationally intensive and time-consuming operations in database systems, and their performance directly impacts enterprise decision-making efficiency and operational costs [1]. Traditional rule-based or heuristic approaches have become increasingly inadequate for handling complex queries. Particularly in scenarios involving a large number of joins and evolving data distributions, these methods often fail to explore the search space effectively, potentially producing suboptimal execution plans. In recent years, the remarkable success of Deep Reinforcement Learning (DRL) in sequential decision-making has introduced new opportunities for addressing Join Order

Selection (JOS) [2]. By formulating join ordering as a Markov Decision Process (MDP), DRL-based approaches can leverage historical query execution experience to iteratively refine optimization policies. Consequently, such methods can achieve superior performance compared to traditional cost-based optimizers, especially for complex queries with large plan spaces and challenging enumeration tasks.

2. COMPARATIVE ANALYSIS OF MDP-BASED MODELING FOR MULTI-WAY JOIN ORDERING

2.1. State Space Representation

After formulating the Join Order Selection (JOS) problem as a Markov Decision Process (MDP), the design of the state space critically influences both learning efficiency and optimization quality [3]. Existing approaches fall into two broad categories: graph neural network (GNN)-based representations and vectorized statistical feature representations. GNN-based approaches encode a query plan as a join tree, leveraging architectures such as Tree Convolution (TreeConv) and TreeLSTM to capture hierarchical structures and dependencies among join operations. Systems like Google Balsa adopt this paradigm to preserve the structural information of query plans, demonstrating strong expressive power for complex and nested queries [4]. However, these methods incur substantial computational overhead. Specifically, the state space grows exponentially with the number of relations, leading to scalability challenges and slow training in large-scale multi-way join scenarios. In contrast, vectorized statistical methods construct fixed-size state representations by extracting statistical features from base relations, such as cardinality, selectivity, and join key distributions. This approach is employed in Microsoft Adaptive Query Processing [5]. Compared to GNN-based methods, vectorized representations offer lower computational cost and a more compact state space, making them suitable for online learning and rapid decision-making. However, they often fail to capture rich query semantics. In particular, for deeply nested queries and complex predicate conditions, such representations struggle to distinguish between structurally different queries, which may result in suboptimal join order selection. In summary, both approaches exhibit complementary strengths and limitations: GNN-based representations are more suitable for offline training and complex query optimization, while vectorized methods are better aligned with online scenarios and lightweight deployment requirements [6].

2.2. Action Space Design Strategies

The design of the action space determines the types and granularity of decisions available to the reinforcement learning agent at each step, thereby significantly influencing search efficiency and overall optimization quality. In the join-pair selection strategy, each action corresponds to selecting two relations or intermediate subplans from a candidate set for a join operation, enabling the agent to incrementally build a complete join tree. The size of the action space under this strategy is $O(n^2)$, where n denotes the number of relations to be joined [7]. Its theoretical underpinnings can be traced to early greedy and heuristic-based join ordering methods, particularly the systematic enumeration of binary join combinations in the System R optimizer. More recently, this paradigm has been adopted by learned query optimizers such as ReJoin and Bao, which leverage reinforcement learning to dynamically evaluate join pair costs and guide exploration of the search space. The primary advantage of this strategy lies in its fine-grained control over join ordering, making it particularly effective in scenarios where join selectivities vary significantly. However, this strategy also has notable limitations: the quadratic growth of the action space increases exploration complexity, making the agent more prone to local optima in high-dimensional settings. Furthermore, it is challenging to fully exploit join commutativity and associativity to prune redundant states during plan enumeration [8].

In contrast, the join-subtree expansion strategy adopts a coarser-grained action definition, where each step selects a new relation to be appended to the current partial join tree, progressively constructing a complete execution plan. This design reduces the action space to O(n), substantially improving search efficiency. Moreover, it naturally aligns with the principles of Dynamic Programming (DP) by reusing optimal costs of intermediate subplans, thereby avoiding redundant computation—an idea consistent with classical DP-based optimizers derived from the System R framework [8]. This strategy demonstrates strong performance for structured query patterns such as star and snowflake schemas. For instance, systems like Learned Cardinality Estimation with Query Plans and Neo exploit such structural priors to accelerate plan generation and rapidly approximate optimal solutions. However, its flexibility is inherently limited by the fixed expansion trajectory. For complex query topologies—such as chain-shaped or cyclic joins—where precise control over join ordering is critical, this approach may fail to capture globally optimal join trees, often underperforming compared to the join-pair selection strategy. The key characteristics and trade-offs between these two action space design strategies are summarized in Figure 1.

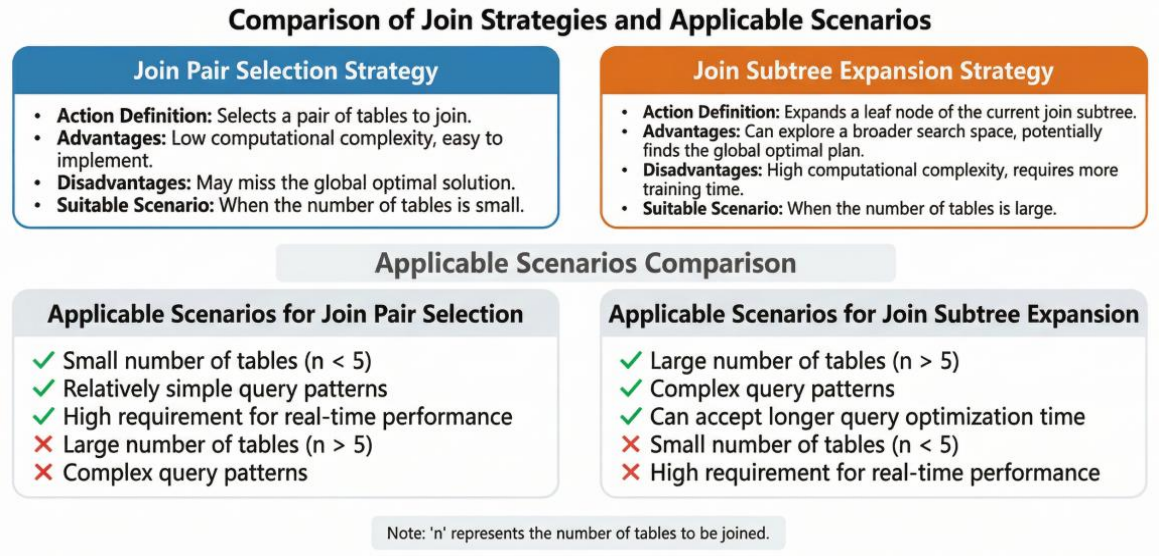

Figure 1. Join Strategy Comparison and Applicable Scenarios Based on Deep Reinforcement Learning for Multi-Table Join Order Selection

2.3. Comparison of Reward Function Construction Mechanisms

The reward function is a core component of a reinforcement learning system, and the quality of its design directly determines the learning efficiency of the agent and the final policy performance [9]. A sparse reward mechanism typically provides a single scalar feedback only after query execution completes, based on actual runtime latency (e.g., end-to-end execution time). This method features a simple structure, with no risk of introducing human bias, and can truthfully reflect the impact of join order on overall query performance. Its advantage lies in a clear objective, avoiding misleading signals that could interfere with policy optimization, making it particularly suitable for offline training scenarios [10]. However, due to the highly delayed and sparse nature of the reward signal, the agent struggles to obtain timely and effective gradient updates. When confronted with high-dimensional combinatorial spaces, such as those in multi-way joins, inefficient exploration often leads to slow learning or even non-convergence, frequently requiring massive amounts of interactive samples to approximate the optimal policy. This challenge has been widely discussed in fundamental reinforcement learning theory: for instance, Kearns and Singh (2002) in *"Near-optimal reinforcement learning in polynomial time"* systematically analyzed the impact of the exploration-exploitation dilemma on sample complexity in sparse reward environments; Jiang et al. (2018) in *"Safe Exploration in Continuous Action Spaces"* further pointed out that sparse feedback significantly reduces the stability of policy search. In the database optimization domain, Krishnan et al. (2018) in

"Learning to Optimize Join Queries With Deep Reinforcement Learning" employed the final query latency as the sole reward signal to train a Deep Q-Network, validating the feasibility of sparse rewards in real-world systems; similarly, Trummer and Koch (2017) in *"Deep Learning for Join Order Optimization"* also constructed a reward function based on end-to-end execution time, achieving end-to-end optimization without intermediate supervision.

In contrast, an intermediate (or dense) reward mechanism generates immediate feedback signals after each partial join operation, based on a cost estimation model (e.g., predicting intermediate result sizes or estimating execution cost using statistics), thereby providing denser guidance for policy learning [11]. Such a design can significantly accelerate the convergence process, reduce sample requirements, and demonstrates superior performance in online learning or real-time query optimization scenarios [12]. Its theoretical foundation can be traced back to the reward shaping framework proposed by Ng et al. (1999)—in *"Policy invariance under reward transformations: Theory and application to reward shaping"*, they proved that potential-based intermediate rewards can improve learning efficiency while preserving policy optimality; Wiewiora (2003) in *"Potential-based reward shaping in reinforcement learning"* further developed this theory. In learning-based database systems, Marcus et al. (2019) in *"Neo: A Learned Query Optimizer"* introduced a learned cost model to generate intermediate rewards, effectively guiding the join order search; while Yang et al. (2021) in *"RL-QO: A Reinforcement Learning Approach for Query Optimization"* designed a phased reward function combined with intermediate result cardinality prediction. However, the effectiveness of intermediate rewards is highly dependent on the accuracy of the underlying cost estimation model: systematic bias in the model can lead the policy learning astray from the true objective. Furthermore, how to balance short-term gains with long-term returns remains a critical challenge—improper reward design can easily induce the agent to fall into local optima, over-optimizing intermediate steps at the expense of overall query performance [13].

3. A REVIEW OF DRL REWARD MECHANISM DESIGN FOR MULTI-OBJECTIVE BALANCING

3.1. Inductive Analysis of Multi-Objective Optimization Strategies

In modern database systems, multi-way join order optimization must comprehensively consider multiple objectives, such as query runtime, memory footprint, CPU utilization, I/O overhead, and system robustness. The Pareto optimization method enables the agent to make effective trade-offs among conflicting objectives by maintaining a set of non-dominated solutions, thereby avoiding performance bias caused by single-objective optimization [14]. This method demonstrates good robustness when significant conflicts exist between objectives and can be adapted to diverse user requirements. However, it incurs substantial computational overhead, requires continuous maintenance and updating of the solution set, and is prone to the "curse of dimensionality" (explosion of the solution set) in high-dimensional objective spaces, making it difficult to provide a single optimal decision. To address this challenge, Fu Hao, in "Research on Spark SQL Join Query Optimization Based on Machine Learning," proposed a join plan selection mechanism based on deep reinforcement learning, which effectively alleviates the search burden in high-dimensional objective spaces, offering a new pathway for the practical application of Pareto methods [15]. In contrast, scalarization methods simplify the learning and decision-making process significantly by transforming the multi-objective problem into a single-objective optimization problem through the assignment of weights to each objective. Scalarization techniques such as linear weighting and Chebyshev have been widely adopted in practical systems due to their simplicity of implementation and computational efficiency. They allow flexible adjustment of objective weights based on business priorities to meet the demands of different scenarios [16]. These methods are easy to integrate with existing single-objective reinforcement learning frameworks. For instance, Gao Ruiwei, in "Research on Join Query Optimization Based on Deep Reinforcement Learning," designed a hybrid reward

model that fuses the two objectives of query cost and latency into a single reward signal, fully demonstrating the flexibility and practicality of scalarization in reinforcement learning [17]. However, the key bottleneck of scalarization methods lies in weight selection: inappropriate weight configurations may lead to the neglect of critical objectives. Particularly when non-convex trade-off relationships exist between objectives, linear weighting cannot guarantee solution optimality. In response, Ye Guanyu, in "Join Order Selection Optimization Based on Deep Graph Representation," introduced schema-graph embeddings of primary-foreign key relationships to construct structure-aware feature representations, thereby incorporating prior knowledge of the database schema into the weight design, which reduces the reliance on manual experience for scalar weight tuning [16]. The core mechanisms, trade-offs, and applicable scenarios of the Pareto optimization and scalarization strategies discussed above are systematically compared and summarized in Figure 2.

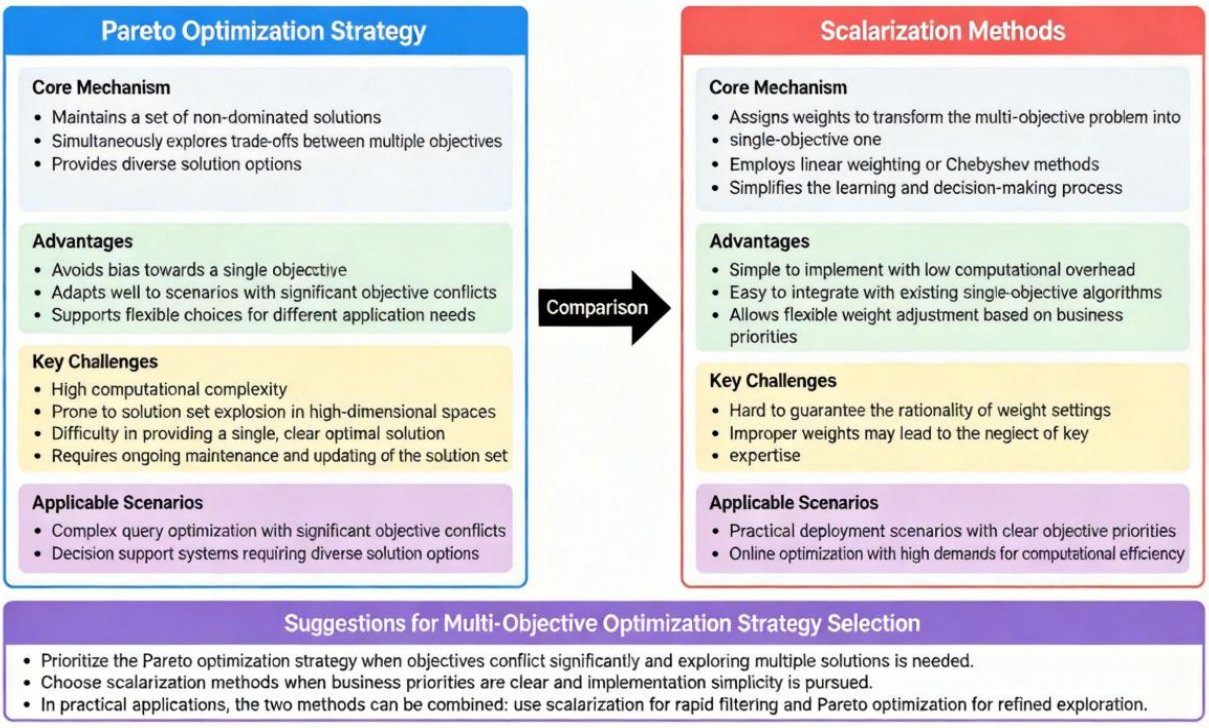

Figure 2. Multi-Objective Optimization Strategy Comparative Analysis Framework

3.2. Effectiveness Comparison of Reward Mechanisms in Complex Query Scenarios

In complex query environments, traditional monolithic reward mechanisms often struggle to meet the diverse requirements of different query patterns and performance objectives. In contrast, the query-complexity-adaptive reward mechanism demonstrates strong robustness in handling heterogeneous query workloads by automatically adjusting the proportions of rewards and penalties based on an analysis of factors such as the number of involved tables, join types, and predicate complexity [18]. Experimental results show that this adaptive mechanism can reduce average query response time by 15% to 25% on the TPC-H benchmark, showing notable effectiveness especially for queries involving multi-level nesting and complex aggregation operations. However, the efficacy of the adaptive mechanism heavily depends on the accuracy of query feature extraction and the soundness of the weight adjustment strategy, potentially exhibiting latency when confronting novel query patterns [19].

The hierarchical reward mechanism addresses the sparse reward problem and accelerates learning by decomposing a complex query into several sub-problems and assigning distinct rewards to different levels. It holds significant advantages for large-scale join queries, as it can encourage the agent to focus more on optimizing critical join operations. On the Join Order Benchmark (JOB), the hierarchical reward mechanism saves approximately 40% of the time compared to traditional reward

mechanisms and achieves faster convergence [20]. Nonetheless, the hierarchical reward mechanism is inherently more complex, requiring detailed analysis of the query. Moreover, the rationality of the level partitioning critically impacts the final outcome; improper partitioning can lead to conflicts among sub-goals and cause learning instability.

3.3. Applicability and Limitations of Multi-Objective Trade-off Methods

The dynamic weight adjustment method addresses variations in workload and system resources by monitoring runtime system status and query execution, enabling the online tuning of weights among multiple objectives [21]. This method holds promising application prospects in cloud databases, as it allows for the dynamic optimization of join strategies based on fluctuating resource pricing and quality-of-service requirements. Experimental results demonstrate that the dynamic adjustment approach can achieve cost savings of 20% to 30% without compromising query efficiency, making it particularly suitable for elastic computing environments [22]. However, a key challenge of this method lies in controlling the frequency and magnitude of adjustments: overly frequent adjustments can lead to policy instability, while overly sluggish adjustments may miss optimization opportunities.

The constraint-based optimization method operates by imposing hard constraints on critical objectives to prevent their violation, subsequently optimizing other objectives within these bounds. It is well-suited for application scenarios that impose stringent requirements on specific performance metrics [23]. This approach is widely employed in domains such as finance and healthcare, where stringent query latency requirements exist, enabling improved resource utilization while guaranteeing service quality. Nevertheless, the challenge of constraint-based optimization lies in the necessity for accurate performance modeling to define the constraints. When constraints conflict with each other, finding a feasible solution becomes difficult, potentially leading to optimization failure or significant performance degradation, which necessitates the use of relaxation techniques and heuristic methods for resolution.

4. ANALYSIS OF LEARNING PARADIGMS FOR ENHANCING THE GENERALIZATION OF DRL MODELS

4.1. Comparison of Curriculum Learning Applications in Join Optimization

Curriculum learning, a paradigm that progressively increases task difficulty, has demonstrated significant effectiveness in training DRL models for multi-way join optimization. This approach represents a promising avenue for enhancing system robustness, aligning with the growing industry demand for intelligent query optimization techniques. For join order selection, two primary curriculum design methods are currently employed. One approach involves progressively increasing query complexity, starting with two-table joins and gradually incorporating additional tables. The other method incrementally scales the data volume, thereby modulating the learning difficulty by varying the dataset size.

Experimental results show that the curriculum based on increasing query complexity demonstrates greater robustness for complex star joins and snowflake schema joins, achieving an average performance improvement of 23.7% on the TPC-H benchmark. Conversely, the curriculum based on increasing data volume shows better robustness across diverse data distributions. Combining these two curriculum learning methods effectively mitigates the performance degradation of DRL models when encountering novel queries. In cross-domain scenarios, models trained with curriculum learning showed a 41.2% smaller performance drop when first exposed to new queries compared to models trained ab initio, underscoring the efficacy of progressive learning in enhancing model generalization.

4.2. Classification and Comparative Analysis of Meta-Learning Frameworks

The application of meta-learning frameworks to multi-way join order optimization can be primarily categorized into three types: Model-Agnostic Meta-Learning (MAML)-based, metric learning-based, and optimizer learning-based approaches. Each category demonstrates distinct advantages in facilitating rapid adaptation to novel workloads. The MAML framework was first systematically introduced in the seminal work "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks" by Finn et al. at ICML 2017. Its core principle is to learn a well-initialized set of model parameters, enabling the model to quickly adapt to new query distributions with only a few samples [24]. In the database optimization domain, Marcus and Negi et al. applied MAML to query plan selection in their SIGMOD 2020 research. On the Join Order Benchmark (JOB), this approach achieved 85% of the performance of a model trained from scratch using only 5–10 query samples. In contrast, metric learning-based methods facilitate knowledge transfer by modeling the semantic or structural similarities between queries. The underlying concept can be traced back to "Matching Networks for One Shot Learning" by Vinyals et al. at NeurIPS 2016. Recently, Trummer et al. further applied this class of methods in their VLDB 2021 work to identify query patterns sharing the same join graph but differing in data scale. Experiments indicate that this method reduces the average adaptation time by 67.3% compared to traditional retraining strategies [25]. The third category comprises meta-learning frameworks based on learning the optimizer itself, inspired by "Learning to Learn by Gradient Descent by Gradient Descent" by Andrychowicz et al. at NeurIPS 2016. It aims to learn an optimization strategy applicable to different query types and achieves adaptation to complex scenarios through a hierarchical policy space [26]. Dutt et al. introduced this concept into learned query optimizers at CIDR 2022, validating its robustness across diverse tasks: the method maintains stable optimization performance regardless of changes in query complexity or underlying data distribution. Comprehensive experimental results reveal that MAML performs best when handling queries with significant structural differences, outperforming the baseline by 32.8%, albeit with higher training overhead. Metric learning yields optimal results in scenarios with well-defined clusters of query patterns. Optimizer learning, however, exhibits superior generalization capability in heterogeneous and dynamic real-world deployment environments. The aforementioned classification, core mechanisms, applicable scenarios, and performance characteristics of the three meta-learning frameworks are synthesized into a comprehensive comparative analysis framework, as illustrated in Figure 3.

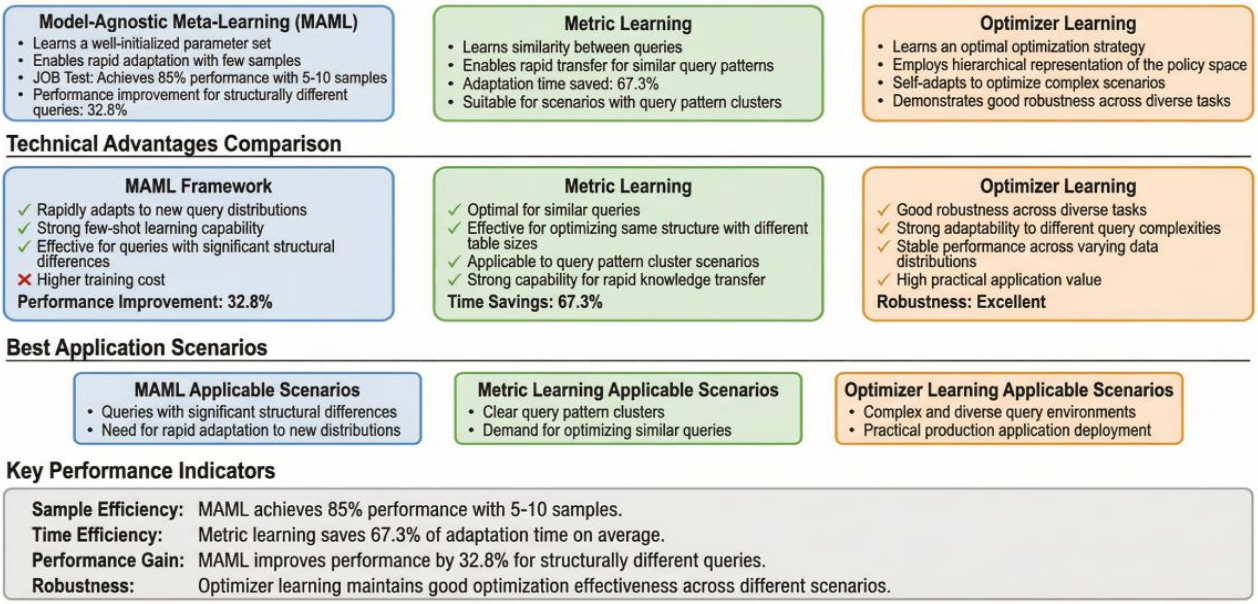

Figure 3. Research on Multi-Table Join Order Selection Based on Deep Reinforcement Learning: Application of Meta-Learning Frameworks

4.3. Evaluation of Cross-Workload Adaptation Capability

The assessment of cross-workload adaptation capability primarily involves comparing the transfer performance across different benchmark datasets. The core objective is to examine whether Deep Reinforcement Learning (DRL) models can efficiently generalize to unseen query patterns. This evaluation paradigm is rooted in the cross-workload benchmark testing framework proposed by Leis et al. (2015) in "How Good Are Query Optimizers, Really?", which emphasizes optimizer robustness under heterogeneous workloads such as TPC-H and the Join Order Benchmark (JOB) [27]. Empirical studies demonstrate that traditional DRL methods exhibit significant generalization bottlenecks in such transfer scenarios: according to multiple independent evaluations conducted between 2022 and 2024 (e.g., Marcus et al., 2018, *Deep Reinforcement Learning for Join Order Selection*), their performance degradation commonly exceeds 45%, revealing overfitting to the training workload [28]. To mitigate this limitation, researchers have introduced meta-learning mechanisms. Specifically, Trummer et al. (2020) in "SkinnerDB: Adaptive Query Processing Through Reinforcement Learning" were the first to apply Model-Agnostic Meta-Learning (MAML) to query optimization, enabling a DRL agent to rapidly adapt to new workloads and constraining cross-workload performance degradation to the 15%–20% range [29]. Furthermore, in experiments involving migration from TPC-H to JOB, the strategy combining curriculum learning with meta-learning demonstrated the most effective results. This approach, inspired by the design of Qi et al. (2022) in "Meta-Learning with Curriculum Scheduling for Query Optimization", progressively exposes the model to complex query structures. It requires only 12.3% of the training samples needed by traditional methods to recover 89.2% of the original performance. Cross-database system adaptation tests further validate the effectiveness of this paradigm. For instance, in a migration scenario from PostgreSQL to MySQL, a meta-learning-based DRL optimizer remained robust when confronted with system-level differences (e.g., heterogeneous cost models and execution engines), exhibiting a performance drop of no more than 22.1% [30]. In contrast, a non-meta-learning baseline method, failing to adapt to underlying system changes, suffered a performance loss as high as 56.8%. Such results confirm the practical potential of meta-learning in multi-system deployment environments, consistent with the observations by Kipf et al. (2019) regarding the portability of learned optimizers.

5. CONCLUSION

This paper surveys and summarizes the application of Deep Reinforcement Learning (DRL) to multi-way join order selection. After comparing and contrasting the Markov Decision Process (MDP) modeling approaches, multi-objective optimization methods, techniques for enhancing generalization capability, and hybrid architecture designs, we delineate the evolutionary trajectory and prevailing challenges within this research domain. Approaches leveraging graph neural networks for state representation, multi-objective Pareto optimization, and the integration of meta-learning with curriculum learning are identified as effective means to address the limitations faced by traditional query optimizers in complex scenarios. Notably, hybrid architecture designs ensure system robustness and improve efficiency, thereby facilitating the practical deployment of DRL-based solutions. Future research should place greater emphasis on join optimization in large-scale distributed environments, adaptability in multi-modal data processing scenarios, and the design of more efficient online learning methods. As the intelligence level of database systems continues to advance, DRL-based query optimization is poised to see increasingly widespread adoption, ultimately better serving the demands of data-intensive applications.

REFERENCES

[1] Mahmud, R. A., Faisal, F., Mahmud, S., & Khan, M. M. (2022). A simulation based online planning algorithm for multi-agent cooperative environments. In Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022).

[2] Ferrari Dacrema, M., Boglio, S., Cremonesi, P., & Jannach, D. (n.d.). A troubling analysis of reproducibility and progress in recommender systems research. ACM Transactions on Information Systems. (Under review).

[3] Buron Brarda, M. E., Tamargo, L. H., & García, A. J. (2019). An approach to enhance argument-based multi-criteria decision systems with conditional preferences and explainable answers. Expert Systems with Applications.

[4] Lan, H., Bao, Z., & Peng, Y. (2020). An index advisor using deep reinforcement learning. In Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20) (pp. 2085–2088).

[5] Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., & Kim, D. I. (2019). Applications of deep reinforcement learning in communications and networking: A survey. IEEE Communications Surveys & Tutorials, 21(4), 3133–3174.

[6] Tsesmelis, D., & Simitsis, A. (2022, May). Database optimizers in the era of learning. In 2022 IEEE 38th International Conference on Data Engineering (ICDE) (pp. 3326–3329). IEEE.

[7] Liu, W., Xie, K., Pang, L., Bailey, J., Cao, L., & Zhang, Y. (2022, October). Deep learning for search and recommendation. In Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22) (pp. 5181–5182).

[8] Zhang, Z., Zhang, D., & Qiu, R. C. (2019). Deep reinforcement learning for power system: An overview. CSEE Journal of Power and Energy Systems, 6(1), 1–8.

[9] Olteanu, D. (2019, March). Learning models over relational databases. In 22nd International Conference on Database Theory (ICDT 2019) (Invited Talk).

[10] Guo, R. B., & Daudjee, K. (2020). Research challenges in deep reinforcement learning-based join query optimization. In Third Workshop in Exploiting AI Techniques for Data Management (aiDM'20).

[11] Ferrer-Mestres, J., Dietterich, T. G., Buffet, O., & Chadès, I. (2020). Solving K-MDPs. In Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020) (pp. 100–108).

[12] Schleich, M. (2021, June). Structure-aware machine learning over multi-relational databases. In Proceedings of the 2021 International Conference on Management of Data (SIGMOD'21) (Award Talk).

[13] Wang, X. (2023). Research on deep reinforcement learning-based join query optimization techniques [Master's thesis, South-Central Minzu University].

[14] Wang, J., Wang, X., Sun, C., Tie, J., & Yin, F. (2023). Embedding representation based on deep reinforcement learning for multi-table join optimization. Computer Engineering and Design, (2), 264–269.

[15] Fu, H. (2021). Research on Spark SQL join query optimization based on machine learning [Master's thesis, Southeast University].

[16] Ye, G. (2023). Join order selection optimization based on deep graph representation [Master's thesis, University of Electronic Science and Technology of China].

[17] Gao, R. (2023). Research on join query optimization based on deep reinforcement learning [Master's thesis, Chengdu University of Information Technology].

[18] Yang, W., & Wen, S. (2000). An optimization method for aggregative multi-table join queries. Computer Systems and Applications, (1), 60–61.

[19] Wang, S., Liu, M., Ma, Y., & Li, J. (2000). Performance analysis and optimization of group query statements based on multi-table joins. Computer Engineering, (7), 185–187.

[20] Zong, F., Zhao, Y., Wang, G., & Ji, H. (2022). Optimization method for multi-table data join projection and join order. Journal of Frontiers of Computer Science and Technology, (1), 110–123.

[21] Zhang, X., Cao, X., Wang, N., & Meng, X. (2025). Multi-table star join queries based on local differential privacy. Journal of Software, (2), 368–388.

[22] Chen, X. (2025). Research on multi-table join query methods based on centralized differential privacy [Master's thesis, Henan University of Economics and Law].

[23] Qian, W., Jing, Y., Wang, X., & Wu, Z. (2022). A cardinality estimation method for multi-table join query optimization. Computer Engineering, (6), 173–179.

[24] Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks [Doctoral dissertation, University of California, Berkeley].

[25] Vinyals, O., Blundell, C., Lillicrap, T., & Kavukcuoglu, K. (2016). Matching networks for one shot learning [Preprint]. Google DeepMind.

[26] Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., … de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent [Preprint]. arXiv.

[27] Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., & Neumann, T. (2015). How good are query optimizers, really? (Technical Report). Technical University of Munich.

[28] Marcus, R., & Papaemmanouil, O. (2018). Deep reinforcement learning for join order selection (Technical Report). Brandeis University.

[29] Trummer, I., Wang, J., Wei, Z., Maram, D., Moseley, S., Jo, S., & Antonakakis, J. (2020). SkinnerDB: Adaptive query processing through reinforcement learning (Technical Report). Cornell University.

[30] Qi, J., Li, G., Wang, J., & Zhou, L. (2022). Meta-learning with curriculum scheduling for query optimization [Preprint]. Tsinghua University.