Journal of Information Systems Engineering and Management
2025, 10(59s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Hybrid Honey Bee Mating and Deep Reinforcement Learning-Based Adaptive Query Plan Optimization

Laradj CHELLAMA[1], Youcef OUINTEN[2], , Benameur ZIANI[3]
[1]Amar Telidji University, Laghouat, Algeria, l.chellama@lagh-univ.dz
[2,3]Amar Telidji University, Laghouat, Algeria

ARTICLE INFO

Received: 26 Dec 2024

Revised: 14 Feb 2025

Accepted: 22 Feb 2025

ABSTRACT

Query plan optimization is a crucial element in the architecture of relational database management systems (DBMS), seeking to identify an ideal execution plan by reducing the overall execution time of queries. Our study employs a novel paradigm, specifically deep reinforcement learning (Deep RL), a subfield of machine learning that integrates reinforcement learning (RL) with deep learning to enhance query optimization methods, which constitute a complete NP problem. In this work , our contribution consists in combining the Honey-bee Mating optimization (HBMO) algorithm to efficiently explore the search space of query plans (stochastic search inspired by the behavior of bees) and adapt DRL algorithms to demonstrate their efficacy. The idea of our approch is to break the problem down into two complementary phases: the Exploration (Global Search) phase uses the Honey Bee Mating Optimization (HBMO) algorithm to find a number of "good" candidate plans by widely exploring the space of possible query plans. The Learning and Adaptation phase uses a Deep Reinforcement Learning (DRL) agent to learn how to choose the best plan among the candidates suggested by HBMO in the first phase.Synergy is produced by hybridization: DRL contributes adaptive intelligence that HBMO alone cannot, while HBMO shrinks the vast research space for DRL. We employ the Proximal Policy Optimization (PPO) method as a model-free approach. Our modest experiments on the IMDB benchmark revealed a progressive gain in Latency (15 to 25%) and a drastic reduction in physical I/O (30-40%), demonstrate that the DRL agent learns to prioritize the most effective plans.

Keywords: Query Optimization,HBMO Deep Learning, TPC-H benchmark.

INTRODUCTION

Optimization is covered by a number of fields, such as computer science, engineering, energy, economics, and medicine. Finding the ideal values for different decision variables in order to solve an optimization problem is the fundamental goal. The goal of an optimization problem is to minimize or maximize a suitable decision-making process, which is frequently modified to allow for approximation techniques. A fundamental aspect of decision-making is selecting from a variety of options. The outcome of this decision is the identification of the "optimal" option among all the options. The query optimizer's goal is to automatically identify the best ways to execute user-provided declarative SQL queries. A query execution plan (QEP), which outlines a strategy for carrying out the query, is produced by the query optimization process. For certain queries, current database management systems (DBMSs) still choose less-than-ideal execution strategies. The goal of this article is to address this problem by outlining and illustrating a novel approach to query execution optimization.

Our theoretical and practical contributions are based on hybridation approches [14],[15] by combining the Honey bee Mating optimization algorithm with deep reinforcement learning for query optimization, which is accomplished by using a learned representation to incrementally construct queries through feature encoding. The idea of our approch is to break the problem down into two complementary phases: the Exploration (Global Search) phase uses the Honey Bee Mating Optimization (HBMO) algorithm to find a number of "good" candidate plans by widely exploring the space of possible query plans. The Learning and Adaptation phase uses a Deep Reinforcement Learning (DRL) agent to learn how to choose the best plan among the candidates suggested by HBMO in the first phase.Synergy is produced by hybridization: DRL contributes adaptive intelligence that HBMO alone cannot, while HBMO shrinks the vast research space for DRL.

This article's following sections are organized as follows: A comprehensive review of recent studies on query optimization problems is given in Section 2. The methodology and experimentation are described in Section 3, including the dataset and methods used. The results gathered and the subsequent discussion are covered in Section 4. The conclusion of the paper is presented in Section 4.

OBJECTIVES

As a result, our endeavor's main goals can be summarized as a reduce user query latency by making the best use of system resources, integrate AI techniques, particularly deep reinforcement learning, into the query processing architecture by simulating the DRL taxonomy's PPO and combined with Honey Bee Mating optimization algorithm.

LIERATURE REVIEWS

The literature, which falls into two categories—traditional optimization and learning-based approaches—has cited several publications that are relevant to our study. For the traditional optimization, [1] describes how System R chooses access paths for both simple (single relation) and complex queries (such as joins),introduced a number of optimization methods, including the use of dynamic programming for bottom-up join tree creation. It was also the first instance of SQL implementation. [2] use the traditional cost model to determine the best plan for a given query by generating different strategies using cardinality estimations, these estimations are based on assumptions and database statistics that might or might not be accurate. Poor execution plans result from incorrect cardinality estimation computations or invalid assumptions. [3] propose a Parametric query optimization (PQO) attempts to solve optimization problem by exhaustively determining the optimal plans at each point of the parameter space at compile time. Most DBMSs use histogram-based techniques as part of their cost model to summarize the data of tables to perform efficient selectivity estimations [4].

For the Learning-Based Query Optimization, recently, several deep reinforcement learning (DRL) based approaches propose using neural networks to construct a query plan. They demonstrate that efficient query plans can be found without exhaustively enumerating the search space [5]. [6] proposed an execution plan recommendation system based on similarity identification between SQL queries. Learning Optimizer [7], is introduced as a comprehensive way to repair incorrect statistics and cardinality estimates of a query execution plan by monitoring previously executed queries and computes adjustments to cost estimates and statistics that may be used during future query optimizations. A novel cardinality estimation approach [8] with the use of machine learning methods that is using query execution statistics of the previously executed queries to improve cardinality estimations and increases the DBMS performance for some queries by several times or by several dozens of times. ReJOIN model [9] focuses on the Join order selection problem by applying deep reinforcement learning techniques. In this model, the agent learns to maximize the reward through continuous feedback with the help of an artificial neural network. Neo (Neural Optimizer) presented in [10], uses a supervised learning model to guide a search algorithm through a large and complex space. Neo assumes the existence of a sample workload which consists of a set of queries that is considered a representative of the total workload. BAO (the Bandit optimizer) presented in [11], is a learned component that sits on the top of an existing query optimizer in order to enhance query optimization rather than discarding the traditional query optimizer. Bao component learns to map the query to the best execution strategy for the query. Then, upon receiving a query, the query optimizer generates multiple plans according to different strategies where the learned model is expected to choose the best query plan given the possible strategies.

METHODOLOGY AND EXPERIMENTS

An overview of the setup needed to carry out the experiments indicated by our research questions is given in the following sections. We hope to offer information that will help ensure the reproducibility of our evaluation. Creating a training and test dataset is the first step in any machine learning task. Given that JOB, which is based on the IMDB database, is being used for benchmarking, it is imperative that our training dataset accurately represents the test dataset. For training and testing, we exclusively used JOB workload in our early trials. we opted to use a new IMDB dataset proposed by Kaiwen Wang in his work [12], this dataset contains 3300 queries consisting of 100 queries for each of the 33 Join Order Benchmark (JOB) models which is composed of 28 tables and 143 attributes, to produce a big dataset. We tested this dataset by executing the different requests via the PostgreSQl 16.1 DBMS on windows 10 to confirm and display the estimate of cardinalities and intermediate result cardinalities using the EXPLAIN statement with JSON format.

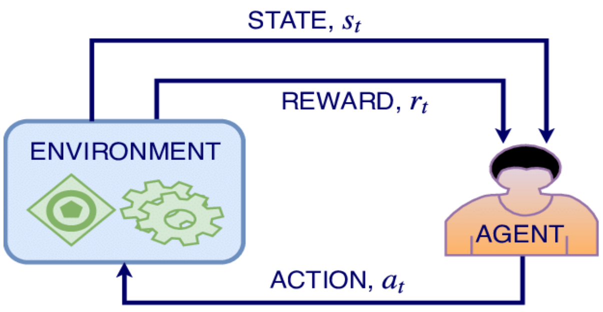
1. APPROCH DESCRIPTION

The Complete Hybrid process initializes with a new complex request, the optimizer first launches the HBMO module which explores the plan space and returns a set P = {$Plan_1$, $Plan_2$, ..., $Plan_n$} of (n) promising candidate plans. Then the DRL agent comes into play, it observes the current state (**S)** of the SGBD system.Based on (S), the agent uses its neural network to select the ($Plan_i$) plan in set P that maximizes the anticipated reward.The request is executed with ($Plan_i$) and its actual execution time is measured. This measure is converted into a reward (R). The couple (State **S**, Action ($Plan_i$), Reward **R**, New State **S'**) is stored in a replay memory to train the neural network. The agent is learning and improving continuously.

2. QUERY ENCODING AND OBSERVATION STATE

RL learning algorithms need an environment in which they can interact with the MDP as explained in figure [1] ,we describe all the components of the agent environment that are necessary to solve a Query Optimization problem with reinforcement learning.

Figure 1: Reinforcement learning problem



An observation represents the state in which the agent currently is. Since we want to provide as much information as possible to our learning algorithms, we formulate query optimization as a fully observed RL problem, the database and the queries be encoded in such a way that the representation can be learned by the RL algorithms. The encoded observation serves as input for a neural network (NN) in order to learn from the observations.

We adopted Krishnan et al.'s [13]technique for the encoding, which uses each database field as a separate feature. Every digit in a binary 1-hot vector that represents a state corresponds to a database column. The vector's size is equal to the total number of columns in all tables. In our case, the observation vector's size (7) is equal to concatenation of query encoding (28 tables) and current join columns (143 columns).

$$obs\_size = 2 * num\_columns + num\_base\_table \quad (1)$$

The agents take actions from all base tables, to construct the action space we take all combinations of all base tables. In our case, the size is equal to 28 tables. A reward is defined as the negative costs of a query plan based on the cardinalities and intermediate results.

3. TRAINING PROCESS

A) PPO ALGORITHM

PPO makes use of the vectorized architecture, an effective paradigm that has a single learner that gathers data and picks up knowledge from multiple environments. PPO initializes a vectorized environment (envs=4) that runs environments either sequentially or in parallel by leveraging multi-processes. The vectorized architecture loops two

Journal of Information Systems Engineering and Management
2025, 10(59s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

phases, the rollout phase and the learning phase, for the first one, the agent samples actions for the (envs=4) environments and continue to step them for a fixed number of steps (num_steps). During these steps, the agent continues to append relevant data in an empty list, in the Learning phase, the agent learns from the collected data in the previous phase of length (envs*num_steps). PPO can estimate value for the next observation and calculate the advantage and the return (returns), both of which also has length (envs *num_ steps).

The weights of hidden layers use orthogonal initialization of weights with scaling (np.sqrt(2)) and the biases are set to 0. Actor-Critic separate architecture which is a type of reinforcement learning algorithm that combines aspects of both policy-based methods (Actor) and value based methods (Critic).This hybrid approach is designed to address the limitations of each method when used individually. PPO uses a simple multilayer perceptron (MLP) network. The critic network architecture consisting of three layers of 512 neurons and using GELU as the activation function. We use Adam Optimizer with epsilon parameter equal to (e^{-5}) and learning rate equal to ($8e^{-4}$).

RESULTS AND DISCUSSION

1. TRAINING PERFORMANCE OF THE PPO ALGORITHM

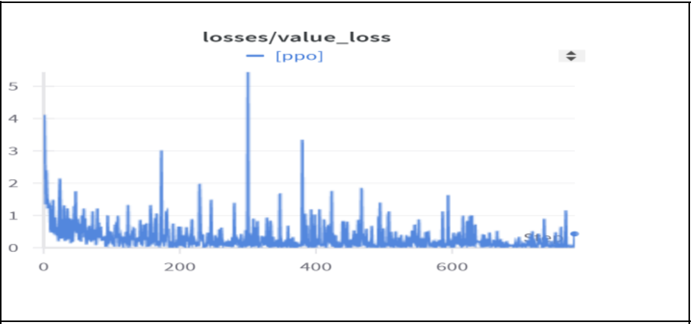
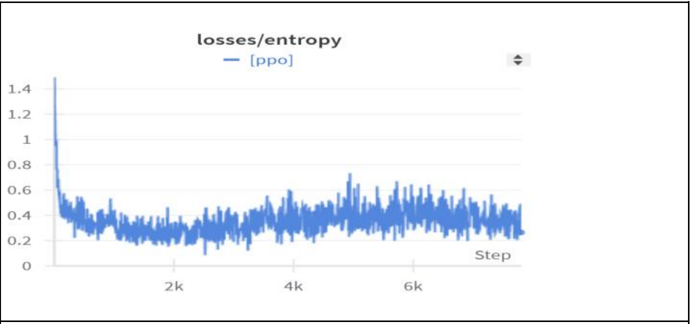
	

Figure 2:PPO value losses	Figure 3:PPO entropy parameter

The graph in Figure 2:PPO value losses corresponds to the model's ability to predict the value of each state, it shows an increase as the agent learns, then decreases once the reward stabilizes.Since, from Figure 3:PPO entropy parameter the model's decisions are random, it can be seen that it decreases slowly during a training process, indicating successful learning.

2. TRAINING PERFORMANCE OF THE HER ALGORITHM

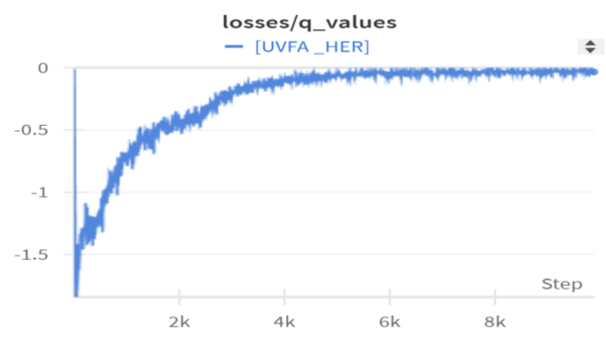


Figure 4: HER q-values losses

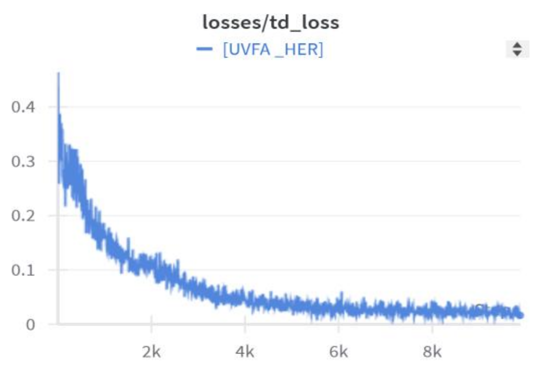


Figure 5:HER td_loss

The graph in Figure 4: HER q-values losses represents a series of values called “losses/q values” according to the “Step”. The values appear to fluctuate significantly, with peaks of about -0.5 and troughs of about -1.5. This significant variability likely indicates that the learning model or strategy is undergoing significant changes during the various stages. It seems to show the evolution of some metrics related to the training of a model or agent in a dynamic environment. he graph in Figure 5:HER td_loss represents the evolution of the metric according to the step for the “[UVFA HER]” model. This metric measures the loss (or error) of the model during its training. It can be observed that the values of this metric start at a high level, around 0.4, then gradually decrease over the training stages. This downward trend indicates that the model learns and becomes more and more efficient to minimize this specific loss. However, there is also great variability in values, with frequent peaks and troughs. This suggests that the model drive is not linear and has phases of progress and setback, probably related to hyper parameter adjustments. Overall, this graph allows following the evolution of the model performance during its training, focusing on the minimization of the loss ‘td loss’.

3. PERFORMANCE OF THE HYBRID APPROCH (1)

Table 1: Performance Metrics

Performance Metrics	Baseline (PostegreSQL)	Our Approch	Gain
Average Latency (Execution Time)	100 % (reference)	75 - 85 %	**15-25 %**
Average I/O Consumption (Physical Reads)	100 % (reference)	60- 70 %	**30-40 %**
Success Rate (Chosen Plan = Best Plan)	N/A (static)	> 90 %	

CONCLUSION

Relationship query optimization, which entails determining a suitable execution strategy, is covered in this work. We propose applying machine learning techniques, such as deep reinforcement learning, to this NP-complete problem. Because of its enormous query volume and the variety of joins it contains. Our study proposes a new approach by exploiting Deep Reinforcement Learning (or Deep RL) a hybrid field combining reinforcement learning and deep neural networks to address the problem NP-complete of the optimization of query plans. We contribute to this field by combining the bee mating optimization algorithm (HBMO), which explores in a stochastic and bio-inspired way the space of query plans, with modern methods of Deep RL in order to increase its efficiency and adaptability.Our method is based on a division of the process into two synergistic phases, a global exploration phase, where HBMO identifies a set of high-quality candidate plans by efficiently traversing a vast research space and a contextual adaptation phase, where a Deep RL agent learns to select the most efficient plan among these candidates based on the current system state.

This hybridization allows to conjugate the exploratory force of HBMO which considerably reduces the decision space and the dynamic adjustment capacity of the Deep RL. We chose the Proximal Policy Optimization (PPO) method, a model-free approach, to drive adaptive decision making. Preliminary experiments conducted on the IMDB database showed notable gains: a reduction in latency from **15 to 25%**, and a decrease in physical input-output of **30 to 40%**. These results indicate that the Deep RL agent is indeed learning to prioritize the most efficient execution plans, thus validating the potential of our hybrid approach for adaptive optimization of complex queries. To sum up, we plan to carry out additional research using different datasets in order to confirm these results and develop a model-based strategy, particularly with regard to the Imagination Augmented Agents (I2A) from the second category.

REFRENCES

[1] Selinger, P. G., M. M. Astrahan, et al. (1979). Access path selection in a relational database management system. Proceedings of the 1979 ACM SIGMOD international conference on Management of data.

[2] Bruno, N. and S. Chaudhuri (2002). Exploiting statistics on query expressions for optimization. Proceedings of the 2002 ACM SIGMOD international conference on Management of data.

Journal of Information Systems Engineering and Management
2025, 10(59s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

[3] Bizarro, P., N. Bruno, et al. (2008). "Progressive parametric query optimization." IEEE Transactions on Knowledge and Data Engineering 21(4): 582-594.
[4] Oommen, B. J. and L. G. Rueda (2002). "The efficiency of histogram-like techniques for database query optimization." The Computer Journal 45(5): 494-510.
[5] Guo, R. B. and K. Daudjee (2020). Research challenges in deep reinforcement learning-based join query optimization.
[6] Zahir, J. and A. El Qadi (2016). "A recommendation system for execution plans using machine learning." Mathematical and Computational Applications **21**(2): 23.
[7] Stillger, M., G. M. Lohman, et al. (2001). LEO-DB2's learning optimizer. VLDB.
[8] Ivanov, O. and S. Bartunov (2017). "Adaptive cardinality estimation." arXiv preprint arXiv:1711.08330.
[9] Marcus, R. and O. Papaemmanouil (2018). Deep reinforcement learning for join order enumeration.
[10]Marcus, R., P. Negi, et al. (2019). "Neo: A learned query optimizer." arXiv preprint arXiv:1904.03711.
[11] Marcus, R., P. Negi, et al. (2021). Bao: Making learned query optimization practical.
[12]Wang, K., J. Wang, et al. (2023). "JoinGym: An Efficient Query Optimization Environment for Reinforcement Learning." arXiv preprint arXiv:2307.11704.
[13]Krishnan, S., Z. Yang, et al. (2018). "Learning to optimize join queries with deep reinforcement learning." arXiv preprint arXiv:1808.03196.
[14]Fathian, Mohammad, Babak Amiri, and Ali Maroosi. "Application of honey-bee mating optimization algorithm on clustering." Applied Mathematics and Computation 190.2 (2007): 1502-1513.
[15]Solgi, Mohammad, Omid Bozorg-Haddad, and Hugo A. Loáiciga. "The enhanced honey-bee mating optimization algorithm for water resources optimization." Water Resources Management 31.3 (2017): 885-901.