Energy-Aware Query Processing: A Case Study on Join Reordering

Ladjel Bellatreche
ISAE-ENSMA
Poitiers, France

Fouad Djellali
ESI, Sidi Bel Abbes
Algeria

Wojciech Macyna
Wroclaw University of Science and Technology
Wroclaw, Poland

Carlos Ordonez
University of Houston
Houston, USA

***Abstract*—Analytic processing systems have been traditionally designed to optimize time performance, leaving energy as a secondary aspect. More recently, during the past decade, there has been a growing interest in addressing energy efficiency of analytics and in particular on query processing (QP), our focus in this article. Numerous solutions, spanning both software and hardware approaches, have been proposed in database systems, but they have important limitations: (i) they were designed for old QP architectures, (ii) they do not consider emerging QP AI trends, such as learned query plans and hybrid QP, and (iii) they lack a well-defined framework with clear steps, which can be applied in a modern data science ecosystem. With such reasons in mind, we introduce a general framework that will help researchers and industry practitioners in addressing energy-efficiency challenges.**

Our framework, named $SATM^2V$, integrates five major steps: (1) assessing public sentiment and alerting analysts on the real impact of data science on decarbonization, (2) conducting "under the hood" energy consumption audits to identify energy-hungry components, (3) turning on/off and tuning parameters of components to understand their contribution to energy savings. (4) developing models and measurement techniques for quantifying energy consumption, (5) developing and executing tactics for energy saving. We then turn our attention to database systems and identify relational join processing as a representative energy consumption example. QP becomes particularly difficult when dealing with queries involving multiple join operations since join ordering is known to be an NP-hard problem, whose optimal solution remains an open problem. We apply our solution framework to the specific case of hybrid QPs, studying the impact of various join ordering optimization techniques on energy efficiency.

Extensive experiments are conducted using the well-known Join Order Benchmark dataset to evaluate the effectiveness and tradeoffs of several query optimization techniques on time and energy consumption.

I. Introduction

The international climate goals established by COP (Conference of the Parties) and the International Energy Agency (IEA) mandate the achievement of net-zero global carbon dioxide (CO2) emissions by approximately mid-century[1]. Additionally, these objectives require the realization of net-zero greenhouse gas emissions by the close of this century. Substantial reductions in emissions, which are becoming increasingly attainable and cost-effective, can be achieved through the enhancement of energy efficiency ($\mathcal{EE}$), the electrification of energy consumption, and the transition to emission-free energy sources. This decarbonization effort encompasses various sectors of our lives, including transportation, buildings, agriculture, and digitalization.

Intensive research initiatives and efforts have been undertaken to study $\mathcal{EE}$ across various objects of different sectors. $\mathcal{EE}$ emphasizes the goal of using the same power to provide better performance or the same service with less energy [1]. Unfortunately, these studies are *imbalanced*. The building sector has made significant strides in terms of scientific research and the accessibility of tools for achieving energy savings, especially when compared to other sectors, such as digitalization. A search on Google Scholar using the query "energy efficiency in buildings" yielded a staggering 3,790,000 entries.

To promote research in digitalization, it is highly recommended to leverage existing initiatives for energy saving in other sectors. This can be achieved by providing a comprehensive framework with well-identified phases for academia and industry to follow. Therefore, we should outline the key dimensions of energy saving in major sectors and adapt them to digitalization, with a particular focus on data science. This will allow us to evaluate and address any aspects that need improvement or refinement.

A. $SATM^2V$ framework

From our in-depth analysis of major publications, our takeaways from the Improvement Project funded by the Sudoe Interreg program, which addresses the challenge of Near Zero Energy Buildings (NZEBs) [2], and our recent studies related to designing green query data processors [3]–[5], we have determined that the $\mathcal{EE}$ of an object in a given sector is typically studied from a combination of the following dimensions:

(1) Public sentiment on the real impact of the object on decarbonisation vary greatly based on factors such as location, demographics, education, and personal beliefs, spanning from optimism to skepticism. Governments and organizations must conduct effective communication campaigns to convince people of the urgent need for decarbonization.

(2) Audis entail a comprehensive evaluation of an object's energy performance and efficiency. Their primary objective is to identify opportunities for energy conservation, cost reduction, and environmental enhancement. Conducting precise and thorough audits can be instrumental in addressing the questions and concerns raised by skeptical individuals regarding the object's impact on decarbonization.

[1]https://www.iea.org/commentaries/the-iea-at-cop26

(3) Tactics are essential techniques used by professionals to control quality of service (QoS) aspects like performance, energy efficiency, and security [6].
(4) Modeling and Measurements of consumed energy of the object involves creating a formal representation of real objects using equations, graphical models, rules, decision trees, examples, neural networks, etc [7]–[10]. This is crucial for building analytical and predictive models, especially for energy consumption. These models are essential for simulating [11] and measuring energy usage in various components of an object [12].

(5) Variability plays a significant role in reducing energy consumption. The $\mathcal{EE}$ of an object is profoundly influenced by variations in its components and external factors [13].

Inspired by the above-depicted factors, we propose a framework called $SATM^2V$ for studying the $\mathcal{EE}$ of objects, encompassing five closely related dimensions (Figure 1). This framework forms the basis for more sustainable and $\mathcal{EE}$ solutions. One notable aspect of this framework is its adaptability to object components. For instance, when using the $SATM^2V$ framework to address $\mathcal{EE}$ in a building, the same tools can effectively be applied to data science. **In the literature, these five dimensions are not commonly studied in an integrated manner**.

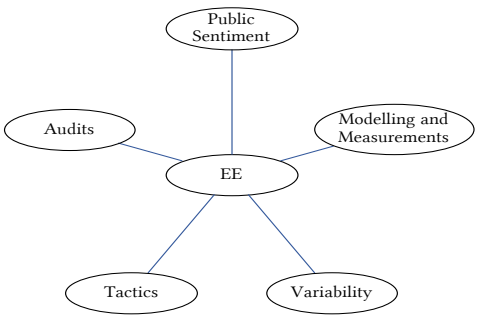

Fig. 1. Dimensions of $\mathcal{EE}$ of an object

In the field of data science, the dimensions described in our framework have been extensively explored, especially in the development of environmentally friendly query processors for use in traditional database servers and data centers. However, the dimension related to the role of variability in object components and external factors in energy efficiency has not received the same level of scrutiny. Some components have been thoroughly analyzed, while others, despite their recognized impact on query processor quality of service, have been disregarded.

In this paper, we investigate the energy efficiency of join ordering. The join operation is a critical component for advanced analytics, techniques like machine learning and predictive modeling. This operation allows the creation of feature-rich datasets for training machine learning algorithms, enhancing predictive model accuracy. In the context of data science applications, the role of join has become even more significant. This heightened importance stems from the requirements of data quality, data integration, and data preparation. Join ordering becomes crucial when query processors employ numerous join operations.

B. Join ordering

Query Processors ($\mathcal{QP}$) have been the subject of extensive research for many decades, primarily focusing on their performance QoS. One critical component of $\mathcal{QP}$ is the query optimizer, which is responsible for identifying the most efficient query execution plans, whether for a single query or multiple queries. This optimization process often leads to the generation of numerous query plans, driven by different strategies for joining various tables or entities.

Numerous existing solutions have been implemented within open-source datastores, demonstrating their efficiency in saving energy and their practicality [14]. However, despite these advancements, these solutions exhibit several shortcomings: (i) any of these solutions were primarily designed for traditional $\mathcal{QP}$s that rely on analytical cost models. (ii) They often do not take into account emerging $\mathcal{QP}$, such as learned $\mathcal{QP}$ [15] and hybrid $\mathcal{QP}$ [16], which require different optimization strategies. (iii) These solutions lack a comprehensive and well-established framework with clearly defined steps that are tailored to the unique characteristics of the data science ecosystem. (iv) They may not deeply analyze the role of variability in $\mathcal{QP}$ elements, such as the choice of join order techniques, in terms of energy conservation.

However, it is noteworthy that, to our knowledge, no existing work has comprehensively investigated the variability of join ordering techniques and their influence on energy savings in a hybrid $\mathcal{QP}$ such as HybridQP system [16].

C. Paper contribution and structure

The contribution of the paper is as follows:
(1) We present a practical application of the $SATM^2V$ framework within the broader data science sector, specifically focusing on data processors ($\mathcal{QP}$s) widely used for data-intensive and workflow applications (Figure 2).

(2) We consider the variability in $SATM^2V$ by analyzing join ordering techniques and their influence on energy savings in data processors. To achieve this, we undertake an in-depth investigation into five distinct join ordering techniques, which include iterative methods, genetic algorithms, PostgreSQL join ordering technique, simulated annealing, and RTOS. These techniques are integrated into the HybridQP system, and we assess their respective energy-saving capabilities using the Join Order Benchmark (JOB). **To the best of our knowledge, it is the first paper that addresses energy efficiency in hybrid query processors.**

Our paper is organized as follows: Section II describes the five pillars of STM^2V framework in the context of data

science. In section III, we present key concepts of the join operation and join ordering solutions. Section IV incorporates the query processors into the framework. We focus on join ordering as an essential and energy-conscious operation. Section V is dedicated to presenting the results of our experimental evaluation. Finally, in Section VI, draw conclusions about our work and provide insights into potential future research directions.

II. STM^2V FRAMEWORK IN DATA SCIENCE

The IEA recognizes the transformative potential of digitalization in the world's energy systems. This sector encompasses various components (Fig. 2) such as Cloud computing, Big Data management, the Internet of Things (IoT), Artificial Intelligence (AI), network technologies, blockchains, and computing power. Given the diversity and heterogeneity of these components, conducting an in-depth analysis is challenging. In this section, we apply the STM^2V framework to data science, an interdisciplinary field that requires various digitalization components.

Several definitions of data science exist; we favor the one provided by Tamer Özsu in his keynote speech at IEEE Big Data Conference 2022 [17].

Definition 1: **Data Science** is a data-based approach to problem-solving by analyzing and exploring large volumes of possibly multi-modal data, extracting from it knowledge and insight that is used for better decision-making. It involves the process of collecting, preparing, managing, analyzing, and explaining the data and analysis results [17].

This definition emphasizes two crucial elements of data science: *data* and the *processes* (Figure 2).

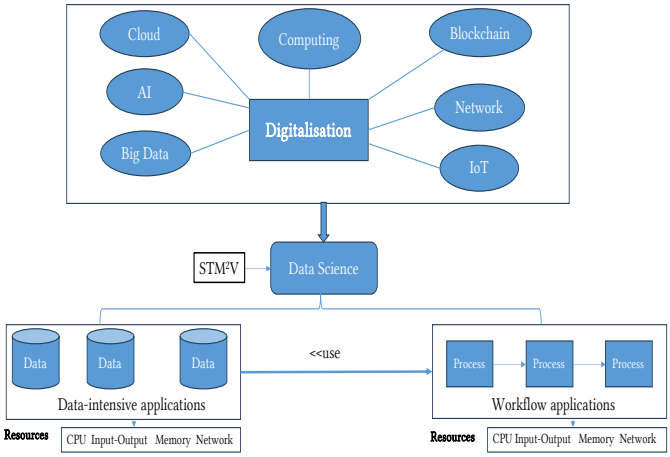

Fig. 2. Digitalisation World

The objective of this section is to describe STM^2V framework from the data science perspective.

A. Public sentiment

Digitalization, in general, and its sub-sector data science, in particular, are sectors that frequently spark contentious debates regarding their role in the process of decarbonization.

a) Data science skepticism

It is essential to note that there is no universally accepted benchmark for monitoring the carbon footprint of elements constituting data science. However, valuable statistics are provided by international organizations. Cloud data centers in charge of storing and processing data, for instance, consumed approximately 205 terawatt-hours (TWh) of electricity in 2020, equivalent to roughly 1% of the world's total electricity consumption[2]. Projections indicate that this consumption is poised to increase to 359 TWh by 2025, signifying a 75% surge from 2020. Network infrastructure components, including routers, switches, and wireless access points, also contribute to energy consumption, as they remain powered on and necessitate *cooling* to maintain peak performance. Heating, ventilation, and air conditioning (HVAC) are responsible for a significant percentage of global CO2 emissions [18].

AI workflow applications that employ extensive machine learning and deep learning models with a high number of features are energy-intensive, covering training, building, and deploying models [19]. Table I provides a concise summary of a recent study investigating the environmental implications of AI workflow applications [19].

The alarming statistics regarding AI energy consumption are compelling the scientific community to develop measurement tools and propose hardware, software, and software-defined hardware infrastructure solutions[3] to mitigate this issue. An urgent area of study that needs to be conducted to mitigate these statistics involves proposing methods for measuring the Return on Investment (ROI) of AI solutions[4].

b) Data science optimism

The defenders of this perspective emphasize the significant positive impact of *data-driven tactics* in enhancing $\mathcal{EE}$ and reducing maintenance costs across various domains. Here are two notable examples: (i) Microsoft's report, published in 2018, highlights the potential of AI in addressing climate change. By using AI to optimize energy consumption, it is estimated that emissions could be reduced by 1.5% to 4.4%, while also boosting global GDP by 3.1% to 4.4% by 2030. (ii) In 2022, Google's DeepMind team introduced the BCOOLER reinforcement learning agent [18]. BCOOLER is designed to enhance the efficiency of cooling systems in data centers. Through AI-driven optimization, BCOOLER achieved remarkable energy savings of 12.7%. This showcases the potential of AI in making existing infrastructure more energy-efficient.

These two examples are indeed intriguing as they provide quantifiable data that highlights the impact of workflow applications on energy savings. For the sake of transparency, it is crucial that such studies incorporate the energy consumption associated with the training process. The evaluation of energy consumption for any AI solution should encompass

[2]https://energyinnovation.org/2020/03/17/how-much-energy-do-data-centers-really-use/
[3]https://semiengineering.com/software-defined-hardware-gains-ground-again/
[4]https://www.oracle.com/a/ocom/docs/artificial-intelligence/idc-ai-ebook-business-transformation.pdf

TABLE I
SOME EXAMPLES ENERGY DEMAND OF AI MODELS: LUCCIONI ET AL. [19]

Model	Number of Features	Data Center PUE	Power Consumption Changes	CO2 equivalent emissions
Gopher	280B	1.08	1,066 MWh	352 tonnes
BLOOM	176B	1.2	433 MWh	26 tonnes
GPT-3	175B	1.1	1,287 MWh	502 tonnes
OPT	175B	1.09	324 MWh	70 tonnes

not only the ongoing energy usage by the target object after deploying the AI models but also the energy expended during the training phase. Several initiatives have been launched to implement machine learning (ML) in ultra-low power systems like TinyML[5].

B. Audits

Conducting a thorough audit of data science requires a comprehensive understanding of its ecosystem. Tamer Özsu identifies the four components of data science ecosystem [17] (Figure 3): **(1)** Big Data Engineering that involves recurring activities for ETL/ELT, data quality, data enrichment, profiling, integration, managing data changes over time, data processing using hundreds of thousands of *periodic data-intensive queries*. **(2)** data analytics that includes *workflow based techniques* such as data mining, machine/deep learning, visualizations. **(3)** data protection (security and privacy issues), and **(4)** data ethics. The life cycle model and its variants may appear to be linear, but in reality, they are cyclical and can be repeated multiple times over the lifetime of a project. The cyclical nature of this life cycle contributes to an increase in the energy consumption of data science applications.

The audit underscores three primary components of data science that are sensitive to energy consumption: data storage, data-intensive applications, and workflow-based applications. These components revolve around data processors, which are at the core of $\mathcal{EE}$ considerations and also have an impact on cooling systems. The join operation stands out from other operations applicable to datasets because it is utilized in various stages, including data preparation, data quality assessment, and model building. As a result, the way the join operation is employed can have a substantial impact on energy savings, particularly if it is not used effectively [20].

C. Tactics

Here we will review important hardware and software tactics for data science components.

a) Hardware tactics

Our review has identified several hardware tactics to conserve energy during data processing. These include: (a) employing microprocessors with lower power consumption (b) utilizing multi-core processors to improve workload distribution and parallelism, (c) incorporating low-power components such as low-power RAM, SSDs, and graphics, (d) optimizing

[5]https://www.eetimes.com/ai-at-the-very-very-edge/

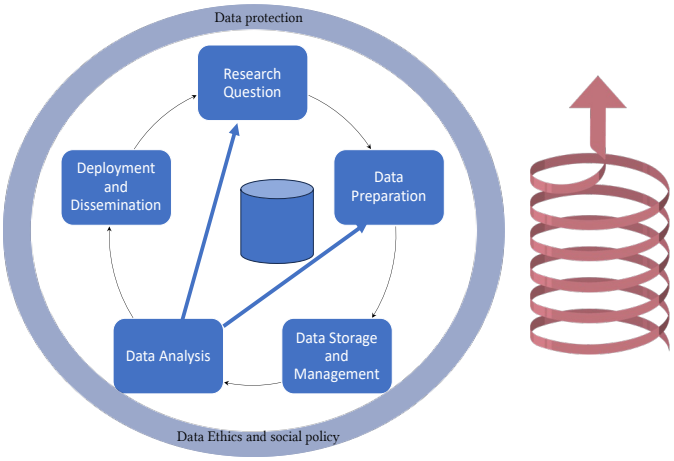

Fig. 3. A data science Lifecycle

cooling systems, like fans and heat sinks, to reduce power consumption, (e) employing high-efficiency power supplies, (f) utilizing energy-efficient processors dedicated to workflow applications, such as the NVIDIA A100 Tensor Core GPU and Google TPU (Tensor Processing Unit), as well as Edge AI Accelerators (g) implementing energy-saving measures in network devices like routers, switches, network interface cards, dynamic power management, packet routing, communication protocols, network resource scheduling, and Dynamic Voltage and Frequency Scaling (DVFS) [21].

b) Software tactics

Several software tactics are available to achieve energy savings. Here are some illustrative examples: (a) query/job scheduling [22], (b) caching strategies [23], (c) data compression [24], [25], (d) parallelism and multithreading [11], (e) optimisation of network communication [26], (f) virtualization and dockerization [27], (g) energy-aware algorithms [28], (h) predictive analysis [5], (i) efficient data processing by optimizing data processing pipelines and reducing unnecessary data transfers citeKatalDC23, (j) energy profiling and monitoring, achieved through software tools that track and analyze energy usage, aiding developers in identifying energy-intensive sections of their applications for targeted optimizations [8], (k) scaling down, particularly in cloud environments, where dynamically scaling down virtual machines and resources during periods of reduced demand conserves energy [29], (l) energy brokering, involving the use of energy brokers in multi-cloud environments to choose the most energy-efficient service

before task assignment, promoting efficient resource utilization [30], (m) employing energy-efficient cooling tools, such as BCOOLER [18].

D. Modeling and Measurements

There has been a wide body of research on energy efficiency modeling of data science components such as data centers, data query processing. The abundance of published papers contributes to having several surveys related to six main topics [7], [9], [31], [32]: computing, storage and data management, network, infrastructures, interdisciplinary, and software and hardware modeling and energy consumption prediction.

As an example, a comprehensive strategy for managing data center energy consumption typically involves four key steps: feature extraction, model construction, model validation, and applying the model to a task like prediction. Feature extraction is the initial step in minimizing a data center's energy usage. It involves quantifying the energy consumption of its components and pinpointing the areas where the most energy is utilized. The next step, model construction, involves utilizing the chosen input features to create an energy consumption model. This is achieved through analysis techniques like regression, machine learning, and more. Following model construction, the next crucial step is model validation, ensuring that the model is suitable for its intended purposes. Finally, the established model can serve as the foundation for predicting the energy consumption of components or systems. These predictions can be leveraged to enhance data center energy efficiency.

E. Variabilitiy

Variability, in the context of software systems or components, is commonly understood as the capacity to adapt to specific contexts [33]. This adaptation may involve making adjustments to the software's structure, behavior, or underlying processes. When examining the energy efficiency ($\mathcal{EE}$) of a query processor, it becomes evident that certain factors of variability, like the join order and the methods used for joining (e.g., hash join, sort-merge join, nested loop join), can have a substantial impact on its energy consumption. Consequently, assessing the energy consumption of query processors requires a departure from conventional software approaches, as these typically treat the data store hosting the query processor as static, without varying its critical elements.

III. Join ordering in query processing

In this section, we introduce the join considered as one of the fundamental operations in data science. The primary goal of a join operation is to combine two entities or tables based on common attributes. Crucial aspects of join operations include their commutative and associative properties. This means that individual joins can be evaluated in any order, enabling the formation of join trees. It is important to notice that different join trees may show very different evaluation performances. A sequence of join operations can easily represented by a join tree that helps in selecting the order in which joins are executed. These trees can take on different shapes, such as deep left trees, deep right trees, bushy trees, and more. The choice of join tree can significantly impact query evaluation performance. The number of possible join trees grows rapidly as the number of join relations increases. The number of binary trees with n leaf nodes is given by $C(n-1)$, where $C(n)$ is the Catalan Number and it is defined as:

$$C(n) = \frac{1}{n+1}\binom{2n}{n} \tag{1}$$

To illustrate this complexity, consider a scenario in which the query processor receives a query involving 21 tables within the context of the Join Order Benchmark. If it employs a left deep tree for query optimization, it must contend with an astonishing *21! (5.1090942e+19)* potential join orders.

This exponential growth poses a challenge in query optimization since assessing all potential join trees is often infeasible. Determining the "most efficient" join tree and join implementation is a complex and challenging problem. The exponential increase in possible join trees as the number of join relations grows makes this problem NP-hard. On the other hand, there are several implementations of join operations, each with its own advantages and trade-offs depending on the size of the involved tables and their data distribution. Common join implementations include nested loop joins, sort-merge joins, and hash joins.

In general, $\mathcal{QP}$s are responsible for the critical task of identifying optimal join strategies and join implementations. This optimization aims to minimize both query execution time and resource consumption.

The problem of join order selection has been widely studied for decades [34]–[36]. Several techniques and algorithms have been proposed in the literature to tackle the problem of join ordering. Three tendencies emerge: (i) some studies used dynamic programming and branch and bound to prune the search space of the join order problem. (ii) the usage of approximate algorithms such as greedy, randomized, genetic strategies [36], Monte Carlos tree search [37] are used, (iii) recently machine learning and deep learning algorithms have been proposed such as RTOS [35], and Rejoin [34]. This interest in join order techniques is entirely justified due to their crucial role in query optimization. The fundamental question that arises is as follows: *Does the join order technique employed by a datastore effectively optimize queries while also conserving energy resources?* To answer this question, let us consider the following example:

Example 1: Let us consider the following query extracted from the Join Order Benchmark [38]. It involves 9 joins.

```
SELECT MIN(n.name)
FROM cast_info ci, company_name cn,
keyword k, movie_companies mc,
movie_keyword mk, name n, title AS t
WHERE k.keyword ='character-name-in-title'
AND n.id = ci.person_id
AND ci.movie_id = t.id
AND t.id = mk.movie_id
AND mk.keyword_id = k.id
```

```
AND t.id = mc.movie_id
AND mc.company_id = cn.id
AND ci.movie_id = mc.movie_id
AND ci.movie_id = mk.movie_id
AND mc.movie_id = mk.movie_id
AND n.name like '%S%';
```

We executed this query with two different join orders: one utilized by the PostgreSQL DBMS and another selected from our exhaustive enumeration of all possible join orders. With the first join order, the query executed in 17 seconds and 606 milliseconds, consuming 0.074 Joules. However, when using the second join order, the query executed much faster, taking only 2 seconds and 104 milliseconds and consuming only 0.044 Joule. This example underscores the significant impact of choosing the right join order on the performance and energy consumption of a query processor.

IV. Query processing of data science in STM^2V framework

In this section, we provide a practical approach for implementing data science query processors within the STM^2V framework. We focus on two crucial aspects of STM^2V: auditing and variability.

A. Auditing of query processors

Four main types of query processors exist: rule-based, cost-based, learned-based, and hybrid (Figure 4). A rule-based query processor applies a set of predefined rules and transformations to the query. These rules are designed to generate an optimized query execution plan that retrieves the desired results in the most efficient way possible [39]. A cost-based query processor relies on the usage of analytical cost estimation models to evaluate different query execution plans. Thanks to heuristics, they choose the one with the lowest estimated cost. These processors have been widely used in traditional databases for decades. They are efficient and provide stable query plans but do not learn from past mistakes [16]. Learned-based query processors have been more recently proposed and leverage machine learning techniques to make query optimization decisions. They learn from historical query performance data and adapt their strategies over time to improve query execution.

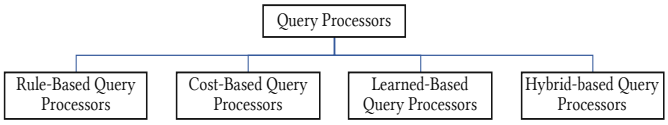

Fig. 4. Taxonomy of Query Processors

It is essential to note that the term "learned query processor" is applied to any $\mathcal{QP}$ that incorporates machine learning or deep learning techniques in at least one of its optimization steps. Several notable examples of learned-based query processors include: Neo System [15] utilizes a learning-based query optimizer that relies on deep neural networks to generate query execution plans. QuickSel adopts query-driven mixture models as an alternative to traditional methods like histograms and samples for adaptive selectivity learning [40]. ReJOIN: ReJOIN introduces a deep reinforcement learning approach to tackle the problem of join order enumeration in query optimization [34]. These processors are effective for static workloads, as they use training data to optimize query plans. However, they may struggle with dynamic workloads [16]. Hybrid query processors have been proposed to overcome the limitations of learned-based processors. They take advantage of both cost-based and learned-based query processors to handle dynamic workloads effectively. HYBRIDQ0 is an example of a hybrid query processor that combines PostgreSQL's cost model with a learning model that predicts execution time and uncertainty [16].

B. Variability of query processors

To investigate the impact of join order variability on query processors, we propose a methodology built upon the system framework of HybridQO. HybridQO is a unique hybrid query processor that merges learned-based and cost-based query (PostgreSQL) processors to determine the best query execution plan.

Its functioning is summarized as follows. When a query is issued, HybridQO encodes it using three vectors: a selection vector, a projection vector, and a join vector. To efficiently manage the join order, HybridQO introduces the concept of join order prefixes. Instead of considering the entire join order, it focuses on a join order prefix (with a length greater than 2). This prefix is important for left-deep plans. These best prefixes can then be recommended to generate alternative join orders.

To illustrate the idea of a prefix, let us take Example 1, which involves 8 joins. A prefix with a length of 2, including two tables from this query, can be implemented in PostgreSQL using a leading hint, as follows:

```
postgres=# /*+ Leading (r t) */ SELECT...
```

HybridQO chooses a join order from the multitude of possible options. Given the high computational cost of evaluating all possible join orders exhaustively, HybridQO employs a learning-based selection approach using the Monte Carlo Tree Search algorithm to select prefixes. Following this, it leverages PostgreSQL to generate complete plans as candidate plans and subsequently utilizes a learning model to select the best candidate plan.

Notably, this model does not only predict plan performance; it also computes the uncertainty associated with the prediction. This *uncertainty* metric gauges the model's confidence in its prediction. Consequently, the model selects the optimal plan that exhibits high performance and low uncertainty. Once the plan is chosen, the query is executed with the selected plan, and the resulting data is employed as a training example to iteratively enhance the model.

The key takeaways from the operation of HybridQO are as follows: (i) the quality of service is primarily tied to query performance and does not take energy consumption

into account. (ii) The join order remains static and does not vary because it relies solely on the Monte Carlo Tree Search algorithm [37].

Our methodology leverages the HybridQO system architecture to tackle these shortcomings. It introduces the energy consumption dimension and variability in the join order selection process. Specifically, we explore four different join order algorithms: Iterative Improvement (II), Simulated Annealing (SA), PostgreSQL's built-in algorithm, and Minimum Selectivity (It consists in building a left linear tree starting from the smallest relationship and joining at each level with the remaining relationship according to the smallest join selectivity factor.).

To support this variability, the initial architecture of HybridQO is extended by two modules: (1) a **join order module** that contains a library of join ordering techniques that substitute the initial join order of HybdridQO, and (2) a **rewriting module**, whose role is to rewrite the initial query according to the selected join order proposed by our algorithms. This rewriting is ensured by PostgreSQL hints (Figure 12).

To accommodate this variability, the initial architecture of HybridQO is expanded with the addition of three modules (Figure 5): (1) **Join Order Module**: it houses a library of join ordering techniques that replace the original join order used by HybridQO. (2) **Rewriting Module**: it is responsible for rewriting the initial query based on the selected join order suggested by the chosen algorithm belonging to the join order library. This rewriting process is facilitated through the use of PostgreSQL hints. (3) **Measurement Module**: it is dedicated to the measurement of our used quality of service, namely, query response time and energy consumption.

Our primary objective in exploring join order variation is twofold: firstly, to analyze and characterize the influence of varying join order on energy savings, and secondly, to question the suitability of using prefixes as a means of achieving energy efficiency.

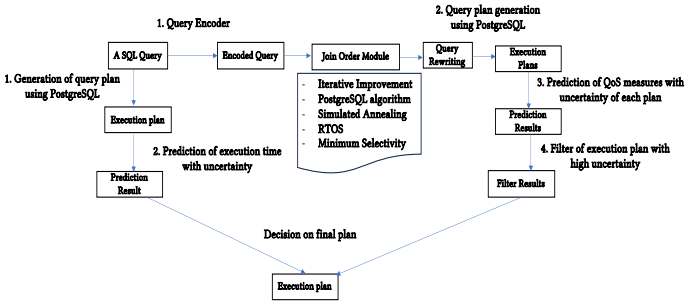

Fig. 5. Energy-aware HybridQO

V. Experimental Study

This section details the experimental studies conducted to evaluate our proposal and to assess the initial decisions made by HybridQO, including the use of prefixes. We begin by outlining the evaluation environment and the measurement tools employed.

Evaluation Environment and Tools: The experiments were performed on a machine equipped with an Intel Core i7 2.8 GHz Quad-Core CPU and 16GB of RAM, running on a 64-bit architecture and MACOS Ventura V13.3.1. All algorithms were implemented using Python v3.7. To measure energy consumption, we utilized the ijoules library, which leverages the Intel Power Gadget API to gather energy data for specific processes. For machine learning modeling, we employed the Scikit-learn library. PostgreSQL DBMS and HybridQO system framework are available at https://github.com/yxfish13/HyperQO.

Datasets: We use the IMDB dataset of JOB having 21 tables and 114 queries available at https://github.com/gregrahn/join-order-benchmark, where several containing more than 10 joins.

We consider two Quality of Service (QoS) metrics in our evaluation: the response time of the employed join order algorithms and the energy consumption associated with these algorithms. Additionally, we assess the energy consumed by the queries themselves.

a) Energy consumed by each join order algorithm

We executed our join order algorithms and meticulously measured their energy consumption for each run. It is important to note that in this study, our objective is to incorporate the training cost into the comprehensive energy calculation. This allows us to assess the return on investment associated with the training within the entire workload. Consequently, when evaluating the RTOS join order, we take into account the energy consumption in both the training and execution phases.

Figure 6 presents a summarized overview of the results, highlighting the significant energy consumption during the training phase of RTOS. These results show also a remarkable stability of PostgreSQL, and Minimum Selectivity in response to an increasing number of joins in queries. However, disparities become evident when considering the iterative improvement and simulated annealing methods, both of which exhibit a noticeable uptick in energy consumption as the number of joins grows.

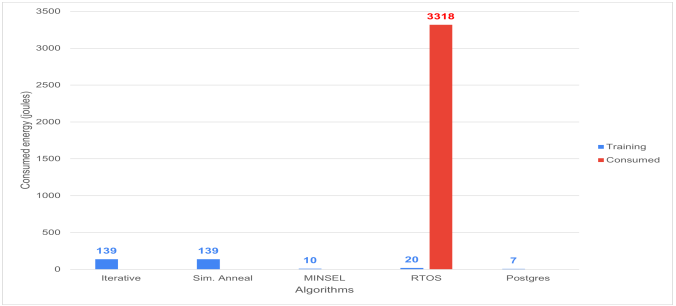

Fig. 6. Energy consumed (by considering training of RTOS) by each join order algorithm

b) Execution time for each join order algorithm

When evaluating the performance of different algorithms in terms of execution time, including training time, we gain additional insights into their efficiency in query processing (Figure 7). Algorithms such as PostgreSQL and Minimum Selectivity continue to outperform others in terms of stability, particularly with regard to execution time. Their ability to maintain relatively consistent execution times despite an increase in the number of joins suggests adaptability to varying workloads. On the other hand, both iterative enhancement and simulated annealing exhibit a trend similar to that observed in energy consumption, where execution time increases as the number of joins escalates. Finally, RTOS stands out as the most time-intensive technique due to the time needed for model training.

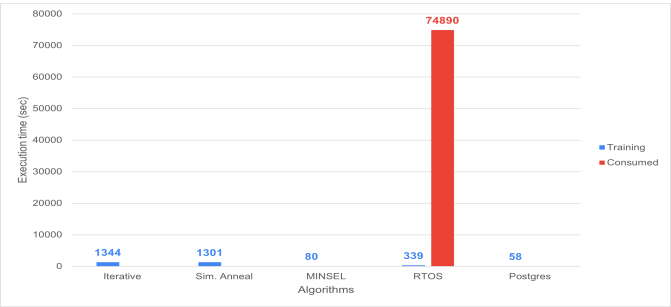

Fig. 7. Response time (including the training time) of each join order algorithm

c) Consumed energy and response time for JOB Workload

In this experiment, we systematically vary the join order algorithms and measure the energy consumption for the JOB workload. For each algorithm, we execute all JOB queries and calculate both the cumulative energy consumption and the cumulative execution time (Figures 8, 9 and 10). This experiment shows a comprehensive and balanced perspective on the performance of each algorithm, taking into account both energy efficiency and execution speed. In terms of energy consumption, the ranking of the best techniques is as follows: 1. RTOS by excluding the training, 2. PostgreSQL, 3. Iterative Improvement, 4. Simulated Annealing, and 5. Minimal Selectivity. PostgreSQL's good performance can be attributed to its use of an exhaustive search when the number of tables is less than 12, and a genetic algorithm when the number of tables exceeds 12. The fact that the number of relations in each query ranges from 4 to 17 positions PostgreSQL's join order technique as a strong candidate for optimizing data manipulation and reducing CPU usage. The obtained results illustrate the return on investment of the RTOS join order technique concerning response time and energy savings. While it is time and energy-consuming during the training phase, it emerges as the most efficient technique for both energy savings and query optimization. Figures 8 and 9 present the energy consumption and the response time of the workload, respectively. Figure 10 shows the behavior of the studied

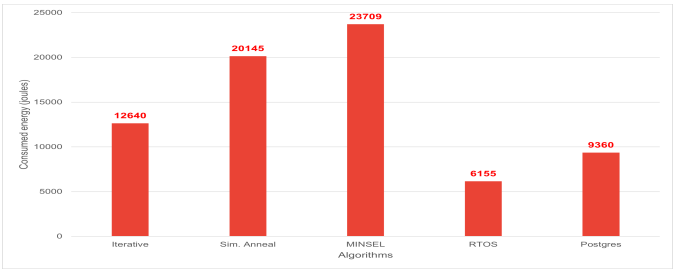

Fig. 8. Energy Consumption Effect of workload by varying join order algorithm

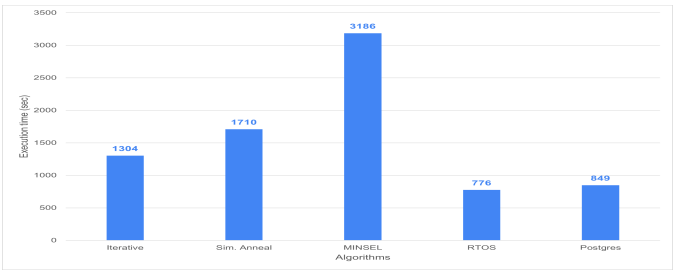

Fig. 9. Response time Effect of workload by varying join order algorithm

join order techniques regarding their energy consumption and response time for all queries of the workload.

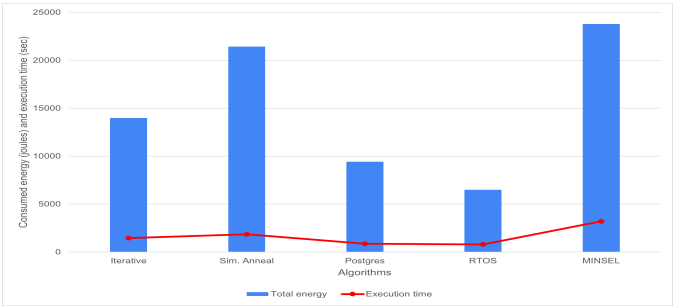

Fig. 10. Energy consumption and response time of workload by varying join order algorithm

In the upcoming experiment, we aim to measure energy consumption for both CPU and RAM. Figure 11 provides an overview of the results. The primary insight gained from this experiment is that the queries used in our study exhibit a significant level of CPU intensity. These results are consistent with previous research on query processors, which has consistently emphasized the energy-intensive nature of CPU usage.

d) Substitution of Monte Carlos Tree Search by RTOS

Recall that HybridQO uses prefixes generated by the Monte Carlos Tree Search technique. We perform an experiment to evaluate the replacement of Monte Carlo Tree Search with RTOS, with the aim of assessing whether the best global join order is also optimal for generating prefixes. To do so, we

Fig. 11. Energy consumption for CPU and RAM for JOB workload

select a subset of queries where the join order generated by RTOS outperforms PostgreSQL. We then extract the prefixes used by RTOS and execute the queries accordingly on PostgreSQL. The obtained results show that having an efficient join order does not necessarily guarantee the quality of an optimal prefix. Based on the results obtained, we recommend the usage of the complete join order generated by RTOS, instead of using prefixes.

Since our experiments clearly identify the superiority and competitiveness of RTOS and PostgreSQL join order techniques, we attempt to evaluate their confrontation of all queries, in order to avoid a join order technique per default. Then we try to evaluate the energy consummated for doing this confrontation. To achieve this, we follow these steps: (1) capture both PostgreSQL and RTOS join orders, (2) generate an execution plan for each join order, and (3) predict the execution time and associated uncertainty for each plan. However, it's worth mentioning that this approach may lead to poorer results in terms of energy consumption since generating and evaluating two execution plans is energy-intensive (Figure 13).

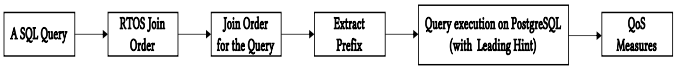

Fig. 12. Substitution of Monto Carlos Tree Search by RTOS

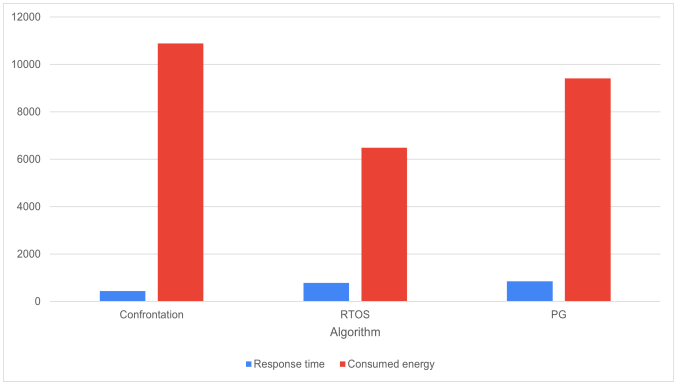

Fig. 13. Effect of confrontation of PostgreSQL (PG) and RTOS

VI. Conclusions

This paper underscored a significant gap resulting from the lack of a comprehensive framework for studying energy efficiency within the digitalization sector, with a particular focus on query processors. These processors play a pivotal role in the development of data science projects, and addressing their energy efficiency is of paramount importance. The creation of such a framework holds the potential to provide invaluable guidance to young researchers and industry professionals. It promotes the idea that energy efficiency in digitalization should concern everyone, and every effort, no matter how small, to save energy should be both undertaken and quantified. This framework enables the categorization of existing research efforts, offers a clearer understanding of ongoing initiatives, and simplifies the initiation and funding of projects related to energy efficiency in digitalization.

To this end, drawing from our experience in participating in the European Project Improvement dealing with the deployment of Nearly Zero Energy Buildings, and by analyzing existing work in various sectors, we introduced a comprehensive framework known as $SATM^2V$. It integrates five key dimensions: (1) assessing public sentiment regarding the real impact of data science on decarbonization, (2) conducting audits of that sector, (3) implementing tactics for enhancing energy efficiency, (4) modeling and measuring the energy consumption of each component of data science, and (5) accounting for the variability of data science components and external factors affecting energy efficiency. Through the application of this framework within the realm of data science, we have unearthed a substantial disparity in public sentiment concerning the role of data science in decarbonization. Our analysis of this dimension underscores the critical need for quantifying this role using comprehensive measurement tools. Moreover, our audit of the data science sector has brought to light the energy-intensive nature of query processors—a crucial insight often overlooked in previous studies that predominantly concentrated on traditional data stores. Furthermore, it is evident that there is a notable lack of diversity in join ordering techniques, even as recent advancements in machine learning-driven approaches to query optimization have emerged.

In response to this, we took decisive steps by considering a hybrid query processing system known as HYBRIDQO. Through the application of various techniques designed to address the join order problem, our experiments challenge the conventional wisdom of using query optimizers without any variation. This approach stands in contrast to previous research on the energy efficiency of query processors, which often neglects the energy costs associated with training machine learning techniques. In our experiments, we meticulously account for and integrate all energy-sensitive components, yielding a comprehensive perspective on the potential for energy savings. Our approach not only questions existing practices but also strives to provide a more thorough understanding of the energy efficiency of query processors.

Currently, we are developing predictive models to estimate and evaluate energy consumption, along with varying hardware components such as the number of cores and processor frequencies.

REFERENCES

[1] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah, "Analyzing the energy efficiency of a database server," in *ACM SIGMOD*, 2010, pp. 231–242.

[2] L. Bellatreche, F. Garcia, D. N. Pham, and P. Q. Jiménez, "SONDER: A data-driven methodology for designing net-zero energy public buildings," in *22nd International Conference on Big Data Analytics and Knowledge Discovery (DaWaK)*, 2020, pp. 48–59.

[3] A. Bouhatous, L. Bellatreche, E. H. Abdelwahed, and C. Ordonez, "The impact of multicore cpus on eco-friendly query processors in big data warehouses," in *IEEE Big Data*, 2022, pp. 4463–4472.

[4] S. P. Dembele, L. Bellatreche, C. Ordonez, and A. Roukh, "Think big, start small: a good initiative to design green query optimizers," *Clust. Comput.*, vol. 23, no. 3, pp. 2323–2345, 2020.

[5] A. Roukh, L. Bellatreche, S. Bouarar, and A. Boukorca, "Eco-physic: Eco-physical design initiative for very large databases," *Inf. Syst.*, vol. 68, pp. 44–63, 2017.

[6] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. SEI Series in 966 Software Engineering, 2013.

[7] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732–794, 2016.

[8] A. Katal, S. Dahiya, and T. Choudhury, "Energy efficiency in cloud computing data centers: a survey on software technologies," *Clust. Comput.*, vol. 26, no. 3, 2023.

[9] B. Guo, J. Yu, D. Yang, H. Leng, and B. Liao, "Energy-efficient database systems: A systematic survey," *ACM Comput. Surv.*, vol. 55, no. 6, pp. 111:1–111:53, 2023.

[10] F. E. Sapnken, M. M. Hamed, B. Soldo, and J. Gaston Tamba, "Modeling energy-efficient building loads using machine-learning algorithms for the design phase," *Energy and Buildings*, vol. 283, p. 112807, 2023.

[11] R. Muralidhar, R. Borovica-Gajic, and R. Buyya, "Energy efficient computing systems: Architectures, abstractions and modeling to techniques and standards," *ACM Comput. Surv.*, vol. 54, no. 11s, pp. 236:1–236:37, 2022.

[12] B. Goel, "Measurement, modeling, and characterization for energy-efficient computing," Ph.D. dissertation, Chalmers University of Technology, Sweden, 2016.

[13] Y. Olivo, A. Hamidi, and P. Ramamurthy, "Spatiotemporal variability in building energy use in new york city," *Energy*, vol. 141, pp. 1393–1401, 2017.

[14] A. Roukh, L. Bellatreche, and C. Ordonez, "Enerquery: Energy-aware query processing," in *ACM CIKM*, 2016, pp. 2465–2468.

[15] R. Marcus, P. Negi, H. Mao, C. Zhang, M. Alizadeh, T. Kraska, O. Papaemmanouil, and N. Tatbul, "Neo: A learned query optimizer," *Proc. VLDB Endow.*, vol. 12, no. 11, pp. 1705–1718, 2019.

[16] X. Yu, C. Chai, G. Li, and J. Liu, "Cost-based or learning-based? A hybrid query optimizer for query plan selection," *Proc. VLDB Endow.*, vol. 15, no. 13, pp. 3924–3936, 2022.

[17] M. T. Özsu, "Data science - A systematic treatment," *Commun. ACM*, vol. 66, no. 7, pp. 106–116, 2023.

[18] J. Luo, C. Paduraru, O. Voicu, Y. Chervonyi, and et al., "Controlling commercial cooling systems using reinforcement learning," *CoRR*, vol. abs/2211.07357, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.07357

[19] A. S. Luccioni, S. Viguier, and A.-L. Ligozat, "Estimating the carbon footprint of bloom, a 176b parameter language model," 2022.

[20] H. Höpfner and C. Bunse, "Towards an energy aware DBMS - energy consumptions of sorting and join algorithms," in *Proceedings of the 21. GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken)*, M. Virgin, A. Peters, and D. Köhn, Eds., 2009, pp. 69–73.

[21] Y. Zhao, W. Zhang, M. Yang, and H. Shi, "Network resource scheduling for cloud/edge data centers," in *39th IEEE International Performance Computing and Communications Conference, IPCCC*, 2020, pp. 1–4.

[22] H. He, H. Shen, Q. Hao, and H. Tian, "Online delay-guaranteed workload scheduling to minimize power cost in cloud data centers using renewable energy," *J. Parallel Distributed Comput.*, vol. 159, pp. 51–64, 2022.

[23] Y. Zhou, S. Taneja, C. Zhang, and X. Qin, "Greendb: Energy-efficient prefetching and caching in database clusters," *IEEE Trans. Parallel Distributed Syst.*, vol. 30, no. 5, pp. 1091–1104, 2019.

[24] I. Raïs, D. Balouek-Thomert, A. Orgerie, L. Lefèvre, and M. Parashar, "Leveraging energy-efficient non-lossy compression for data-intensive applications," in *17th International Conference on High Performance Computing & Simulation, HPCS*, 2019, pp. 463–469.

[25] S. Lee, K. Kim, G. Koo, H. Jeon, M. Annavaram, and W. W. Ro, "Improving energy efficiency of gpus through data compression and compressed execution," *IEEE Trans. Computers*, vol. 66, no. 5, pp. 834–847, 2017.

[26] D. J. Bahadur and L. Lakshmanan, "A novel method for optimizing energy consumption in wireless sensor network using genetic algorithm," *Microprocessors and Microsystems*, vol. 96, p. 104749, 2023.

[27] M. Dinga, I. Malavolta, L. Giammattei, A. Guerriero, and R. Pietrantuono, "An empirical evaluation of the energy and performance overhead of monitoring tools on docker-based systems," in *ICSOC*, Z023.

[28] W.-c. Feng, *The Green Computing Book: Tackling Energy Efficiency at Large Scale*. USA: CRC Press, Inc., 2014.

[29] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu, S. U. Khan, and A. Y. Zomaya, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, no. 7, pp. 751–774, 2016.

[30] B. Aldawsari, "An energy-efficient multi-cloud service broker for green cloud computing environment," Ph.D. dissertation, Liverpool John Moores University, UK, 2018.

[31] J. Wang, L. Feng, W. Xue, and Z. Song, "A survey on energy-efficient data management," *SIGMOD Rec.*, vol. 40, no. 2, pp. 17–23, 2011.

[32] K. Ebrahimi, G. F. Jones, and A. S. Fleischer, "A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities," *Renewable and Sustainable Energy Reviews*, vol. 31, pp. 622–638, 2014.

[33] J. van Gurp, J. Bosch, and M. Svahnberg, "On the notion of variability in software product lines," in *Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2001, pp. 45–54.

[34] R. Marcus and O. Papaemmanouil, "Deep reinforcement learning for join order enumeration," in *aiDM@SIGMOD*. ACM, 2018, pp. 3:1–3:4.

[35] Y. Xiang, L. Guoliang, C. Chengliang, and T. Nan, "Reinforcement learning with tree-lstm for join order selection," in *ICDE*, 2020, pp. 1297–1308.

[36] M. Steinbrunn, G. Moerkotte, and A. Kemper, "Heuristic and randomized optimization for the join ordering problem," *VLDB J.*, vol. 6, no. 3, pp. 191–208, 1997.

[37] C. Browne, E. J. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. P. Liebana, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, 2012.

[38] V. Leis, A. Gubichev, A. Mirchev, P. A. Boncz, A. Kemper, and T. Neumann, "How good are query optimizers, really?" *Proceedings of the VLDB Endowment*, vol. 9, no. 3, pp. 204–215, 2015.

[39] J. C. Freytag, "A rule-based view of query optimization," in *ACM SIGMOD*, 1987, pp. 173–180.

[40] Y. Park, S. Zhong, and B. Mozafari, "Quicksel: Quick selectivity learning with mixture models," in *ACM SIGMOD*, 2020, pp. 1017–1033.