Optimal Index Selection using Optimized Deep Deterministic Policy Gradient for NoSQL Database

V. Sumalatha
Department of Computer Science Engineering, Osmania University, Hyderabad, India
sumala7627@gmail.com (corresponding author)

Suresh Pabboju
CBIT, Hyderabad, India
plpsuresh@gmail.com

Received: 27 August 2024 | Revised: 19 September 2024 | Accepted: 29 September 2024

 DOI: https://doi.org/10.48084/etasr.8832

ABSTRACT

As big data technology has developed, so have complex applications that require increasing resources. The need for high-performance reading and writing increases the usage of NoSQL (MongoDB) databases. As the number of queries in a given amount of time negatively affects the performance of the database, an automated index selection strategy should be used to improve the database performance. This study proposes an Optimized Deep Deterministic Policy Gradient (ODDPG) to select the optimal index. The Adaptive Crocodile Optimization Algorithm (ACOA) is used to improve DDPG's decision-making performance. The ACOA algorithm is used to receive the best action sequences of a DQN. Simulation results showed that the proposed method achieved better results than the existing DDPG model by 2.3% in Average Time Of Query (ATQ) executed, 10% in Query Per Hour (QPH), and 11% in throughput.

Keywords-NoSQL; MongoDB; DDPG; ACOA; QPH

I. INTRODUCTION

Big data can be effectively stored and processed using cloud computing databases, and NoSQL databases are part of these technological advances. NoSQL databases use different data models for different purposes, including key-value, wide-column, and document models. NoSQL is used in modern web-scale databases because of its performance, availability, and scalability. For the past 3 to 4 decades, larger organizations and enterprises have been using conventional databases, such as Relational Databases (RDBs), to store and analyze data. RDBs have a structural model and support SQL (Structured Query Language). Traditional RDBMSs, constructed to obtain structured data, perform poorly when dealing with unstructured data [1, 2]. Thus, HDFS and MongoDB have gained attention in cloud data storage, as they can obtain a larger count of nodes containing complex data and show high fault tolerance and scalability. Therefore, some cloud applications store data in NoSQL databases [3].

NoSQL databases introduce two important issues. The first is that databases are increasingly difficult to provide efficient data management, as using a KnOBD is more complicated for common users [4-6]. The second is their large amounts of resource consumption in calculating statistical properties. These statistical properties are often difficult to obtain in large databases. Index selection should be applied to all possible indexes and workloads, which is a complicated issue [7, 8]. Several techniques have been presented to improve the performance of NoSQL databases. In [3], a spontaneous method (DRLISA) was introduced to achieve index selection in a NoSQL database. DRLISA established the optimal index using a deep reinforcement learning method that was adapted to the dynamic change in workloads. In various workloads, this approach spontaneously obtains the required indices and parameters to increase the database learning process. Experimental results showed the high efficiency of DRLISA in dynamic workloads. In [9], an advanced method for CANDECOMP/PARAFAC (CP) rank selection was proposed, based on deep reinforcement learning. The Deep Deterministic Policy Gradient (DDPG) algorithm was used to automate and improve index selection. This method was used to handle the Decomposition Game (DecG) single-player game structure. The pre-trained model was used to confuse and reassemble different historical data that were given as network inputs. However, these methods still face challenges, such as insufficient robustness.

In [10], security problems were investigated in pattern formats in NoSQL databases. A pattern-based approach can be used to understand other significant but underexplored areas in NoSQL databases. In [11], an optimum index selection method was proposed to choose cattle breeds. The optimum index

depends not only on the breeding values but also on their squares. The characteristics allocated for selection aim to stabilize and maximize the expected total profit for the offspring, without focusing on increasing the short-term process. In [12], cooperative information criteria were used for optimal group selection in air quality forecasting. This study suggested a selection method that relied on cooperative information rules to find optimal forecasting sets from different individual models.

II. OPTIMAL INDEX SELECTION USING OPTIMIZED DEEP DETERMINISTIC POLICY GRADIENT FOR NOSQL

A. Overview

Figure 1 shows the general design of the proposed method, where W denotes the workload and G denotes the index configurations given as input. The input sets are denoted as the state vector S, and A defines the group of actions combined with the state vector. Therefore, in Deep Deterministic Policy Gradient (DDPG) methods, the set of S and A is put as input. Depending on the input sets, the DDPG structure is evaluated and the output is estimated as Q values. Based on the Q values, optimal action sequences are selected using the ACOA algorithm. The ACOA algorithm summarizes discounted return, which examines a fitness function used to select the optimal action sequence. This optimizes storage in the DDPG structure. The optimal action segment received provides input to another stage that proposes an optimized DDPG structure. This reduces the loss function of the output in ODDPG, and every level is upgraded until the optimal index configuration is found.

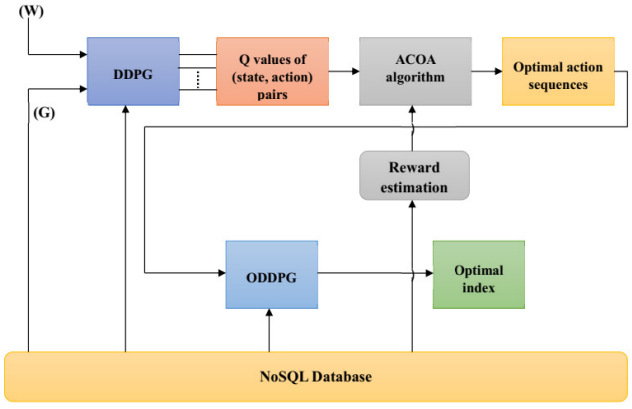


Fig. 1. The overall structure of the proposed approach.

B. Deep Deterministic Policy Gradient (DDPG)

Figure 2 shows the DDPG model. The off-policy data point gets recollected to utilize the Q-function, where the learn policy implies the Bellman equation. In every set, the quality of action is given as $a*(s)$, denoting the optimal action-value function to resolve the problems.

$$a*(s) = argmax\ {}_aQ^* \quad (s,a) \tag{1}$$

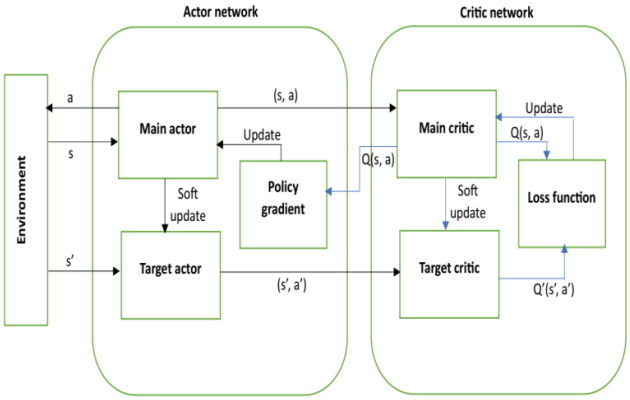


Fig. 2. DDPG structure.

DDPG denotes the learning approximator $Q*(s,a)$ with a learning approximator $\pi*(s)$ evaluated especially in relevant formats of environments in sustained action spaces. Therefore, DDPG is suitable for all environments with sustained action space. It gives immediate action that increases a Q-value. Therefore, a given action space is sustained, and the function $Q(s,a)$ is evaluated in various forms regarding the action arguments. This helps to create an efficient, gradient-based learning step for a set $\pi*(s)$. In the final stage, rather than using a high-quality optimization function several times, $max\,a\,Q(s,a)$ can be obtained using available information.

$$max\,a\,Q(s,a) = Q(s,\mu(s)) \tag{2}$$

A Bellman-based equation combines the optimal actions and the function value of $Q*(s,a)$ as:

$$Q*(s,a) = E_{s`\sim P}\left[r\left(s,a+\gamma\ max\,a`Q*(s`,a`)\right)\right] \tag{3}$$

The symbol $s`\sim P$ denotes the other state $s`$ produced from a probability distribution P expressing the current state s and the action a. The Bellman-based formula is used in the first step of constructing a similar function for $Q*(s,a)$. A similar neural network $Q^{\varphi}(s,a)$ is assumed, with parameter φ and a transition set $D = (s,a,r,s`,d)$, where $d$ represents the state $s`$ terminal. A Mean-Squared Bellman Error (MSBE) function is built to find how well Q^{φ} fits the Bellman-based equation:

$$L(\varphi,D) = E[(s,a,r,s`,d)\sim D]\times$$
$$\left[\left(Q^{\varphi}(s,a) - \left(r+\gamma(1-d)\,max\,a`Q^{\varphi}(s`,a`)\right)^2\right)\right] \tag{4}$$

The important algorithms used to learn a DNN to estimate $Q*(s,a)$ depend on the experiences replay buffer that compares a group of adventures (D). The buffer replay should be large enough to cover a wide area of adventures. The Q-learning algorithm uses a targeted network as shown in

$$r+\gamma(1-d)\,max\,a`Q^{\varphi}\quad(s`,a`) \tag{5}$$

The MSBE loss can be reduced by the function target, and the Q-function is mapped after this reduction. Unfortunately, this goal depends on the way that is used to track. MSBE can be reduced in various ways. The aim is to use a set of access parameters, while in time delay, it reacts with the secondary

network defined as a target network with backward as first. The target network is represented as $\varphi targ$.

$$\varphi_{targ} \leftarrow p\varphi_{targ} + (1-p)\ \varphi \quad (6)$$

where p expresses the high standard form in the middle of 0 and 1, mostly near 1, known as polyak. DDPG deals with network policy that targets the calculation similarly to maximize $Q\varphi_{targ}$.

The target policy-based network is found approximate to the target Q-function: it is reached by taking a polyak average of the policy parameters in search periods. In the same model, Q-learning was used to reduce the MSBE by stochastic gradient descent in the below format:

$$L(\varphi, D) = E_{(s,a,r,s`,d)\sim D}\left[\left(Q_{\varphi}(s,a) - \left(r + \gamma(1 - d)\ Q_{\varphi targ}\left(s`, \mu_{\theta targ}(s`)\right)\right)\right)^2\right] \quad (7)$$

where $\mu_{\theta targ}$ is the target policy.

Therefore, the state-value function varied with an ongoing action space in parameter θ_{μ} can be improved by using a gradient descent. In the derived algorithm, the gradient gets searched using the series form:

$$\nabla_{\theta\mu} J = \nabla_{a^i{}_t} Q\left(s_{i,t}, a_{i,t}|\theta_Q\right)\nabla_{\theta\mu}\mu\left(s_{i,t}|\theta_{\mu}\right) \quad (8)$$

C. Selection of Optimal Action Sequences

The ACOA algorithm [13] can solve optimization problems using swarm intelligence. ACOA can obtain the optimal solution from an information flow where the components get combined to supply data. This algorithm is based on the chasing attitude of crocodiles, which are of two types: ambushers and chasers. Big chasers do not eat fish but send them ashore by lashing their tails violently. Ambushers wait in-depth and try to catch fish. This study combined the ACOA with the OBL for the proposed system. OBL helps in processing better searching capabilities. Then a perfect action sequence is selected to describe formats in ACOA.

1) Initialization

The random solutions for basic populations are initialized using:

$$y = L + rand \times (U - L) \quad (9)$$

where y expresses a random solution, L and U express a lower and an upper bound of the variable, respectively, and $rand$ expresses a random number in $\varphi_{targ}[0,1]$. The group of Q-values is designed to solve the problems. The population P_t expresses the total number of the solution Y at t as:

$$P_t = [y_1, y_2, \dots, y_Y]_t \quad (10)$$

The problem in y_Y is expressed as:

$$y_Y = [Q^i]_Y \quad (11)$$

Therefore:

$$Q^i = \{Q(s_i, a_{i1}), \dots, Q(s_i, a_{iM})\}$$

2) OBL

All original results should be generated using contrast results. A contrast result is expressed as:

$$\overline{y} = a + b - y \quad (12)$$

whereas $y \in [a, b]$ is a real number.

3) Fitness Calculation

The fitness is calculated for every initial solution. The discount returned (R) is expressed as the fitness function. The sequence of actions is the sum of all credits as:

$$R_i = \sum_{m=1}^{M} \lambda^{M-m} r(s_i, a_m) \quad (13)$$

and the fitness function is:

$$Fit_i = Max(R_i) \quad (14)$$

Accordingly, the optimal action sequence (13) is denoted as a result of the largest discount getting a refund. Therefore, the solution becomes better when the steps are updated and discovered. The numerical definition of hunting is shown below.

4) Chasing the Prey

Here, the first step is exciting the chasers. Therefore, the chasers are very close to food, and the first 50% of crocodiles (solutions) team up to chase. The other 50% of the solution includes the ambushers. Comparing both the chasers and ambushers groups, the latter is more away from the food. Therefore, a conclusion is made to find the length between the food and the crocodiles. A chaser is closer to the food, but ambushers may be far away from the prey. The behavior of capturing food is formulated in the following equations.

$$d = \left|y^t_{prey} - y^{i,t}_{chaser}\right| \forall i \quad (15)$$

$$y^{i,t+1}_{chaser} = \left(y^{i,t}_{chaser} - \beta\bar{r}.d\right) \quad \forall id < \alpha \quad (16)$$

$$y^{i,t+1}_{chaser} = \left(\bar{y}_{rand} - \beta\bar{r}.d\right) \quad \forall id \geq \alpha \quad (17)$$

where y^t_{prey} denotes the prey placed in iteration t, $y^{i,t}_{chaser}$ is the i crocodile at iteration t, $\bar{y}_{rand}$ is the random vector location, β is the equal allocation changing within range 0 and 3, r is a random value within the range 0-1, and d is the space interval within the i^{th} chaser and the prey.

Every, chaser moves toward its food with a positive coefficient of (16) if $d < \alpha$. Random search is agreed if $d \geq \alpha$. Thus, the chasers attack their food when it is close to them, otherwise, the random approach is followed.

5) Attacking the Prey

Chasers attack the prey at its final place in the area. Attacked forms are expected using the following equations:

$$d = \left|y^t_{prey} - y^{i,t}_{ambusher}\right| \quad \forall i \quad (18)$$

$$a = \frac{apc + apa + preyposition}{3} \quad (19)$$

$$y^{i,t+1}_{ambusher} = \left(d.cos(2\pi) + y^t_{prey}\right) \quad \forall id < \alpha \quad (20)$$

$$y^{i,t+1}_{ambusher} = \left(y^{i,t}_{a} - \beta\left(A - y^{i,t}_{a}\right)\right) \quad \forall id \geq \alpha \quad (21)$$

where $y_a^{i,t}$ denotes the position of the i^{th} ambusher at iteration t, d is the distance between the ambushers and the prey, apc denotes the chasers' average position, and apa denotes the ambusher's average position. Motion depends on the results of (19). If $d < \alpha$, the ambusher swims over the prey. This depends on the average status of every class of chasers, the ambushers, and the prey. The location of ambushers is upgraded using (20).

6) Termination Criteria

These steps are repeated until finding the optimal solution. Then, the algorithm ends. This optimal action sequence is used to attain an optimal index configuration.

III. RESULTS AND DISCUSSION

Experiments were carried out on a PC with Windows 10, an Intel Core i7 processor, 8 GB of RAM, and the Python programming language. The proposed method was tested using the "Yahoo! Cloud Serving Benchmark" (YCSB). This benchmark contains common data manipulation reports such as reading, upgrading, inserting, scamming, and Read-Modify-Write (RMW). The tests of the proposed method varied in workloads with different datasets: 20% (Read), 40% (20% Read, 20% Insert), 60% (20% Read, 20% Insert, 20% Update), 80% (20% Read, 20% Insert, 10% Update, 30% Scan), and 100% (20% Read, 20% Insert, 10% Update, 30% Scan, 20% RMW).

A. Comparison of DDPG, DQN, and Existing Approaches with and without Optimization

1) Average Time of Index Selection (ATIS)

Figure 3(a) shows the ATIS of the proposed ODDPG. The ODQN achieved an ATIS of 3362 ms, while OQ-learning and ORandom achieved ATIS of 4335 ms and 6504 ms, respectively, for 100% workload. However, compared to ODQN, the proposed ODDPG achieved better ATIS at 2875 ms. Figure 3(b) shows the ATIS of DDPG, which achieved 3082 ms while DQN, Q-learning, and Random achieved ATIS of 3748, 5016, and 7128 ms, respectively.

2) Average Time Query Execution(ATQ)

Figure 4(a) shows the ATQ of the proposed ODDPG with optimized Q-learning, DQN, and Random selection. Due to the optimal action selection using the ACOA algorithm, the ATQ of the proposed method in ODDPG was reduced to 4238 ms with 20% workload and 8234 ms with 100% workload. Figure 4(b) shows a comparative analysis of the ATQ of DDPG. Compared to other models, DDPG achieved better ATQ of 4521 ms for 20% and 8635 ms for 100% workload.

3) Throughput

Figure 5(a) shows a comparative analysis of the throughput of ODDPG combined with optimized Q-learning, DQN, and random selection. ODDPG throughput increased to 10%, 19%, and 54% than that of ODQN, OQ-learning, and ORandom, respectively. Figure 5(c) shows the comparative analysis of the ACT for DDPG. Compared to DQN, Q-learning, and Random, the ACT of the DDPG was reduced to 17%, 39%, and 64%, respectively.

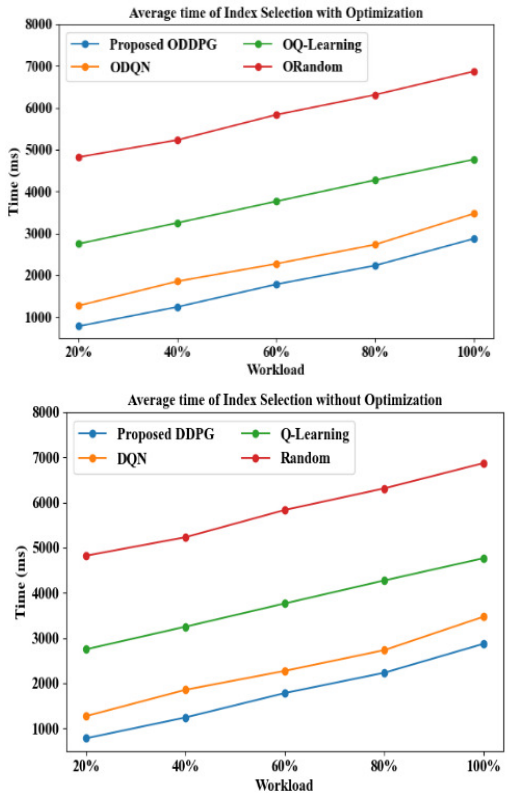


Fig. 3. (a) ATIS with optimization, (b) ATIS without optimization.

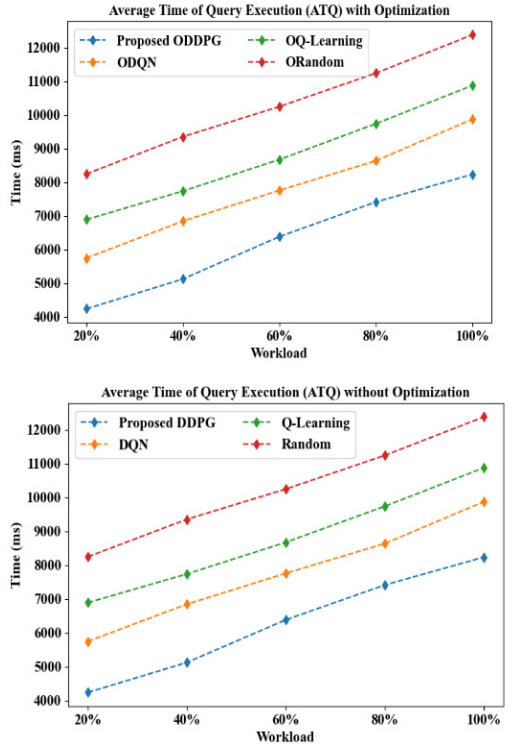
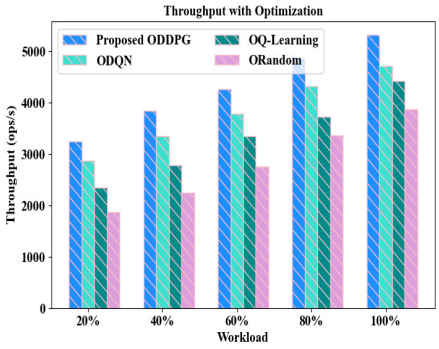


Fig. 4. (a) ATQ with optimization, (b) ATQ without optimization.

(a)



(b)

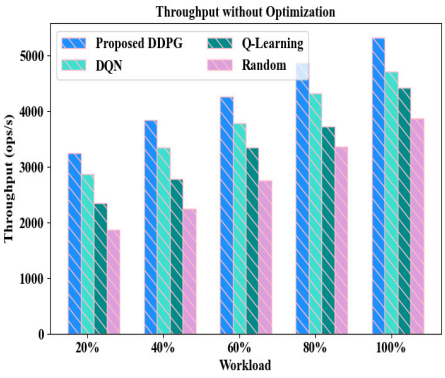
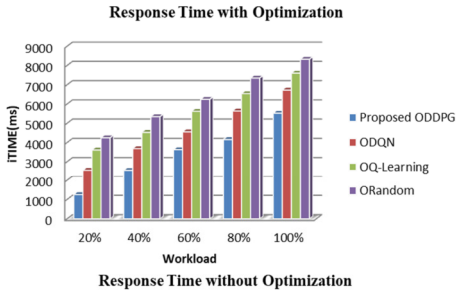


Fig. 5. Throughput with (a) and without (b) optimization.

4) *Response Time*

Figure 6(a) shows the response time of the proposed ODDPG. ODQN achieved a response time of 6724 ms while OQ-learning and ORandom achieved response times of 7602 ms and 8327 ms, respectively, for 100% of the workload. Compared to ODQN, the proposed ODDPG achieved a better response time of 5524 ms. Figure 6(b) depicts the response time of the DDPG at 6198 ms compared to DQN, Q-learning, and Random that achieved 7341 ms, 8124 ms, and 8694 ms, respectively, at 100% workload.

(a)



(b)

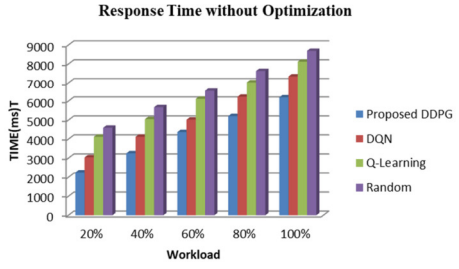
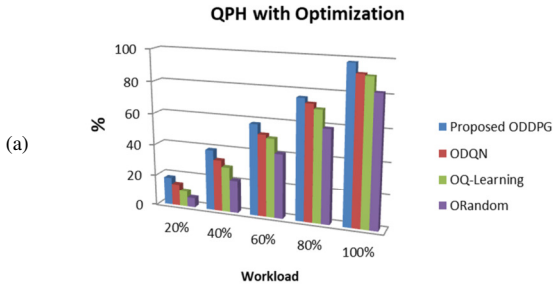


Fig. 6. Throughput with (a) and without (b) optimization.

5) *QPH*

Figure 7(a) shows a comparative analysis of the QPH of the proposed ODDPG with optimized Q-learning, DQN, and Random selection. Due to the optimal action selection using the ACOA algorithm, the QPH of the ODDPG increased to 18% and 98% for workloads of 20% and 100%, respectively. Figure 7(b) shows the comparative analysis of QPH, where DDPG achieved better QPH of 15% and 95% than existing models for workloads of 20% and 100%, respectively.

(a)



(b)

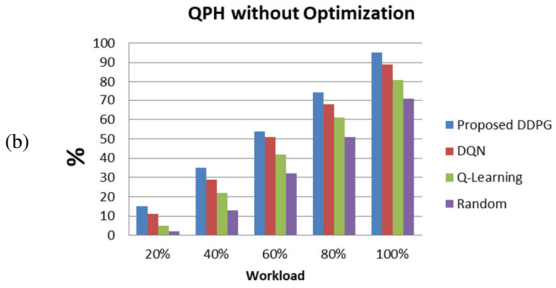


Fig. 7. Throughput with (a) and without (b) optimization.

IV. CONCLUSION

The number of queries in a given amount of time negatively affects the performance of NoSQL databases. Automatic index selection plays a vital role in improving database performance for managing large numbers of queries. This study proposed an optimized DDPG for optimal index selection. Using the ACOA algorithm, DDPG was able to estimate the optimal action format. MongoDB was used to examine the execution of current models. The execution of each method was compared with the conventional DDPG model. The results show that the proposed method increased throughput by 11% compared to the conventional DDPG model. Future studies should focus on enhancing the security of NoSQL database systems during query executions.

REFERENCES

[1] X. Gao and J. Qiu, "Supporting Queries and Analyses of Large-Scale Social Media Data with Customizable and Scalable Indexing Techniques over NoSQL Databases," in *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Chicago, IL, USA, May 2014, pp. 587–590, https://doi.org/10.1109/CCGrid.2014.57.

[2] C. Blanco *et al.*, "Security policies by design in NoSQL document databases," *Journal of Information Security and Applications*, vol. 65, Mar. 2022, Art. no. 103120, https://doi.org/10.1016/j.jisa.2022.103120.

[3] Y. Yan, S. Yao, H. Wang, and M. Gao, "Index selection for NoSQL database with deep reinforcement learning," *Information Sciences*, vol. 561, pp. 20–30, Jun. 2021, https://doi.org/10.1016/j.ins.2021.01.003.

[4] S. Kim, Y. Hoang, T. T. Yu, and Y. S. Kanwar, "GeoYCSB: A Benchmark Framework for the Performance and Scalability Evaluation of Geospatial NoSQL Databases," *Big Data Research*, vol. 31, Feb. 2023, Art. no. 100368, https://doi.org/10.1016/j.bdr.2023.100368.

[5] E. Petraki, S. Idreos, and S. Manegold, "Holistic Indexing in Main-memory Column-stores," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, Melbourne, Australia, May 2015, pp. 1153–1166, https://doi.org/10.1145/2723372.2723719.

[6] N. R. Gayathiri, D. D. Jaspher, and A. M. Natarajan, "Big Data retrieval techniques based on Hash Indexing and MapReduce approach with NoSQL Database," in *2019 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, Sathyamangalam, Tamil Nadu, India, Apr. 2019, pp. 1–8, https://doi.org/10.1109/ICACCE46606.2019.9079964.

[7] P. Ameri, "On a self-tuning index recommendation approach for databases," in *2016 IEEE 32nd International Conference on Data Engineering Workshops (ICDEW)*, Helsinki, Finland, May 2016, pp. 201–205, https://doi.org/10.1109/ICDEW.2016.7495648.

[8] F. Alsayoud and A. Miri, "Index Selection on MapReduce Relational-Databases," in *2015 IEEE First International Conference on Big Data Computing Service and Applications*, Redwood City, CA, USA, Mar. 2015, pp. 302–307, https://doi.org/10.1109/BigDataService.2015.23.

[9] S. Zhang, Z. Li, W. Liu, J. Zhao, and T. Qin, "Rank Selection Method of CP Decomposition Based on Deep Deterministic Policy Gradient Algorithm," *IEEE Access*, vol. 12, pp. 97374–97385, 2024, https://doi.org/10.1109/ACCESS.2024.3428370.

[10] K. Goel and A. H. M. T. Hofstede, "Privacy-Breaching Patterns in NoSQL Databases," *IEEE Access*, vol. 9, pp. 35229–35239, 2021, https://doi.org/10.1109/ACCESS.2021.3062034.

[11] R. Wellmann, "Selection index theory for populations under directional and stabilizing selection," *Genetics Selection Evolution*, vol. 55, no. 1, Feb. 2023, Art. no. 10, https://doi.org/10.1186/s12711-023-00776-4.

[12] Z. Ding, H. Chen, and L. Zhou, "Optimal group selection algorithm in air quality index forecasting via cooperative information criterion," *Journal of Cleaner Production*, vol. 283, Feb. 2021, Art. no. 125248, https://doi.org/10.1016/j.jclepro.2020.125248.

[13] A. R. Balavand, "Crocodile Hunting Strategy CHS): A comparative study using benchmark," *Iranian Journal of Numerical Analysis and Optimization*, vol. 12, no. 2, Sep. 2022, https://doi.org/10.22067/ijnao.2022.71418.1049.