International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Modified Deep Q Optimizer with Updated SARSA for Improving Learning Efficiency through Optimum Resource Utilization in Reinforcement Learning

[1]Dr. Vaqar Ahmed Ansari, [2]Dr. Kamal Shah, [3]Dr. Rohini Patil, [4]Dr. Anil Vasoya, [5]Dr. Payel Saha, [6]Ms. Mary Margaret, [7]Mr. Suresh Rajpurohit

Submitted: 26/11/2023 **Revised**: 07/01/2024 **Accepted**: 19/01/2024

Abstract: In Reinforcement Learning (RL) efficiency of the algorithm is ensured by reducing the cost of learning with maximizing the rewards. In this paper a new technique for RL-based Deep Q optimizers is introduced with Updated SARSA algorithm and newly defined linear cost function. Modified DQ perform significantly faster after learning as it utilize ϒ value for particular epoch instead of ϒmax for the whole dataset .Proposed linear cost model gives wide range of weight parameters "W" where the mean value is always closer to the minimum cost which implies easy to make cluster and train the features. Proposed modified DQ gives 88.2% reduction in variation for relative mean cost for proposed cost-model. With proposed cost model, training execution time for modified DQ has been reduced by 19.35% compared to existing DQ and improves accuracy by 3 to 4% with optimum reward.

Keywords: *Reinforcement learning, Deep Q optimiser, Cost Model, Reward function*

1. Introduction

Machine learning enable algorithms to learn from various patterns. Deep learning takes the support of neural network with multiple layers to automatically extract the primary features from data.[1] Reinforcement learning (RL) uses algorithms that learn from outcomes and decide which action to take for next iteration. After every action algorithm learns from the feedback and as per positive feedback progress further. This approach helps RL to take autonomous decisions. RL elements are Policy, Reward function, Value function and Environment. In RL [2] software agents learn through trial-and-error method and each correct action earns a reward. Deep RL (DRL) [3] is an advance model at the intersection of deep learning and reinforcement learning. It aimed at training agents to make sequential decisions in complex environment. In DRL integration for feature extraction through deep learning and RL's sequential decision-making capabilities empowers agents to learn autonomously various behaviours and development of AI systems. The aim of a RL is to execute the policy which gives maximum rewards and for which calculation of 'Q ' is required. The policy can be created based on the optimum Q value can be achieved using Deep Q Neural Networks (DQN)[4].RL is pivotal advancement where in deep neural network are employed to approximate the Q functions which is best suited for the complex environment like video games and robotics. This paper works on implementation of RL using modified Q learning techniques supported by updated SARSA ensuring an optimum utilization of memory and resources.

[1]*Assistant Professor,St.Francis Institute of Technology*
Email: vaqar@sfit.ac.in Orcid id: 0000-0003-0841-9554
[2]*Professor (IT Dept), Thakur College of Engineering and Technology, Mumbai, India. Email: kamal.shah@thakureducation.org Orcid id: 0000-0002-6369-6960*
[3]*Assistant Professor, Terna Engineering College, Navi Mumbai,India.*
Email: rohiniapatil01@gmail.com Orcid id: 0000-0001-9476-3559
[4]*Associate Professor, Thakur College of Engineering and Technology, Mumbai, India. Email: anilk.vasoya@thakureducation.org Orcid id: 0009-0009-6495-9142*
[5]*Professor Extc Department, Thakur College of Engineering and Technology ,payel.saha@thakureducation.org*
[6]*Assistant Professor , Thakur College of Engineering and Technology, Mumbai*
Mary.margarate@thakureducation.org
[7]*Assistant professor Computer Engineering St Francis Institute of Technology*
****Corresponding author:** kamal.shah@thakureducation.org*
***Corresponding author:** Dr Kamal Shah , IT department Thakur College of Engineering and Technology, Mumbai*

2. Literature Review

To understand the strength of RL and deep neural network [5-6] for designing novel algorithm for optimum 'Q' value calculations various research papers were discussed along with their strength and area of improvement. In [7] Overestimation problem is handled with Deep SARSA and Q learning algorithm. This study was done for global information which has increased the speed of learning process. Study showed cumulative rewards in RL have improved. Value function achieved through neural networks gives better results using flow–shop Scheduling approach. Performance evaluation done using maximum, minimum and mean of various operations. Deep Learning

(DL) and Multi-agent Systems (MAS) which are applied in RL as Deep Reinforcement Learning (DRL) are discussed. [8]

Study [9] discussed about the multiple agents in the same environment ,their behaviour and learning.

Application of DRL various fields like healthcare, fintech, Natural Language Processing (NLP) are discussed with respect to data preprocessing. Many open-ended research problems are also mentioned in [10].In paper [11] approach of Multi Q-learning is discussed. The results shows that returns have improved by 2.5 times than traditional approach. In technical report [12] the hash-join algorithms to a multiprocessor architecture is described. Multiple centralized join algorithms are measured and its performance improvement is observed compared to existing algorithms. Optimization of Q through various approaches like join queries, very long queries and progressive queries are discussed in [13-15]. As some of the RL methods perform poor if values of Q are overestimated.so double Q approach discussed in [16] improves the performance of RL models. Query optimization helps RL to learn faster. In [17-19] query execution time is estimated for various workloads in database. Off policy evaluation methods discussed in [20].

3. Proposed Algorithm

Reinforcement Learning gives maximum efficiency when the agent learns from environment and collects highest rewards. To ensure modifications the proposed methodology is divided into two phases in phase I Modified DQ learning algorithm is proposed where a fitness function helps RL to learn quickly for environment and in phase II SARSA algorithm is modified by taking average value of ' λ' till particular epoch which will ensures better learning speed and then all the modifications are checked with different memories allocations.

3.1 Modified Deep Q Learning Algorithm (MDQL)

Algorithm1 of MDQL is mentioned below.

Algorithm1: Modified Deep Q Learning Algorithm (MDQL)
1: Import libraries of Python
2: Define policy for Reward
3: Define the environment and agent
4: Define a new fitness function and supporting fitness functions
5: As per fitness function calculate the values of Q
6: Calculate the reward function value

To update the existing Q learning algorithm a fitness function is introduced. The aim is to train a bot to find the location using these Environmental Clues. Let an individual be a bit string of a fixed length k with x bits set to 1. The target fitness function (FF) is given as,

$$g(x) = \left[\frac{x}{k}\right] \quad (1)$$

Supporting Fitness Functions becomes,

$$h1(x) = min\,(x, p)$$

and

$$h2(x) = max\,(x, p) \quad (2)$$

Where 'p'= positive integer

The task for the suggested method is to dynamically switch algorithm to the most appropriate current FF basing on the kind of the individuals in the current generation. The method should set the current FF to $h1$first and switch it to $h2$ when the individuals in the current generation reach the switch point. The value of the reward function depends on changes of the fitness of the best individual, which appear after the creation of the next generation caused by the action of the agent. An auxiliary function D_f is defined as per the equation (3)

$$D_f(x_1, x_2) = \{0\ if\ f(x_2) - f(x_1) < 0\ 0.5\ if\ f(x_2) - f(x_1) = 0\ 1\ if\ f(x_2) - f(x_1) > 0 \quad (3)$$

The reward function is shown in equation (4)

$$R(s, a) = D_g(x_s, x_{s'}) + c(D_{h_1}(x_s, x_{s'}) + D_{h_2}(x_s, x_{s'})) \quad (4)$$

Where,

'c' = a real-valued parameter that allows contribution of fitness functions to the reward.

s and *s'* = the previous and the new state respectively,

a= The action that caused transition from s to s'

x_s and $x_{s'}$ are the number of bits set to one in the best

3.2 Modified SARSA Algorithm

SARSA is a State-Action-Reward-State-Action reinforcement learning algorithm [21] It is used to learn a policy for an agent interacting with an environment. In On Policy of RL algorithm, the agent learns the value function according to the current action derived from the existing policy while in Off Policy the agent learns the value function according to the action derived from another policy [20].Q-Learning technique is an Off Policy technique based on greedy approach to learn whereas SARSA technique, is an On Policy and uses the action performed by the another policy to learn the Q value.

Existing SARSA-Bellman Equation is [22]

$$Q(s_{t,}a_t) = Q(s_{t,}a_t) + \alpha(r_{t+1} + \Upsilon max Q(s'_{t+1}, a'_{t+1})) - Q(s_{t,}a_t) \quad (5)$$

Where,

- $Q(s_{t,}a_t) =$ $Expected\ reward\ for\ taking\ action\ a_t\ in\ state\ s_t$
- $r = Actual\ award$
- $s'_{t+1}, a' = Next\ state\ and\ action$
- $\alpha = Learning\ rate$
- $\Upsilon max =$ $Discount\ factor\ calculated\ at\ the\ end\ of\ cycle$

To improve the speed of learning instead of calculating discount factor after each step and to reduce the reward to keep final reward bounded as modified to Υ where value of discount factor is calculated till particular Epochs as seen in Modified SARSA Eq.6

$$Q(s_{t,}a_t) = Q(s_{t,}a_t) + \alpha(r_{t+1} + \Upsilon Q(s'_{t+1}, a')) - Q(s_{t,}a_t) \quad (6)$$

Algorithm2: Modified Q Learning and Modified SARSA (MQLMS)	
1:	Start
2:	Observe the current state
3:	Initialize the $Q(s_{t,}a_t)$to some arbitrary values
4:	Based on value of Ɛ take a decision for exploration or exploitation
5:	In Exploitation state Modified Q value is calculated as per Eq. 6
6:	In Exploration state based on Fitness function values are calculated
7:	Based on the either step 5 or 6 action a_t is selected
8:	Based on action reward is received
9:	Q table is than updated based on reward value for the next state
10:	Repeat steps 4-8 until the set epochs end

This loop continues till the desired Fitness function is achieved with minimum cost and maximum reward. The below figure 1 shows the flowchart of MQLMS methodology.

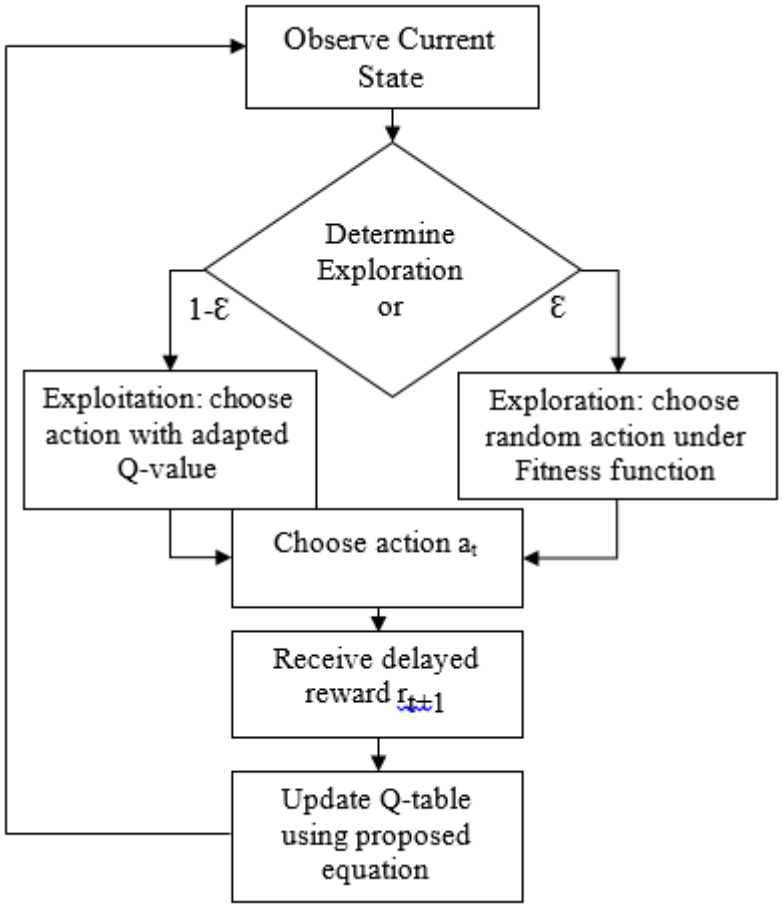

Fig.1: Flowchart of Modified Q Learning and Modified SARSA (MQLMS)

One of the primary objectives of RL is to minimize the cost of learning. To realize the same a novel cost function model is defined.

Cost Model 1: The first cost model [19], uses main-memory database that performs two types of joins: index joins and in-memory hash joins

$$c_{ij}(O)=c(O_l)+\text{match}(O_l,O_r).|O_l| \quad (7)$$

$$c_{hj}(O)=c(O_l)+c(O_r).|O| \quad (8)$$

where

c = the cost estimation function,

$c_{ij}(O)$= cardinality function, and match denotes the expected cost of an index match.

Cost Model 2: In cost model 2 index eligibility is removed and only hash joins and nested loop joins with a memory limit M.

$$c_{join}=\{c(O_l)+c(O_r)+|O| \quad if\ |O_r|+|O_l|\leq M\ c(O_l)+c(O_r)+2(|O_r|+|O_l|)+|O|\ \ if(|O_r|\ c(O_l)+c(O_r)+\left(|O_r|+\lceil\frac{|O_r|}{M}\rceil|O_l|\right).|O_l|)\ \leq\ M^2 \quad (9)$$

The non-linearity in this model are size-dependent. Controlling the size of intermediate relations is important in the optimization problem

Cost Model 3: It is based on Gamma database where the left operator as the "build" operator and the right operator as the "probe" operator [23].

$$c_{nobuild}=c(O_l)+c(O_r)\ -|O_r|+|O| \quad (10)$$

Proposed Cost Model:

Cost function optimization algorithms attempt to find the optimal values for the model parameters by finding the global minima of cost functions. Gradient Descent algorithm makes use of gradients of the cost function to find the optimal value for the parameters.

Hypothesis: $h(x)=W+bx$ (11)

On each iteration t, the cost of the data is found.

Cost Function:

$$C(W,b)=\frac{1}{2t}\sum_{i=1}^{t}\ (h\left(x^{(i)}\right)-y^{(i)})^2 \quad (12)$$

The partial differentiation of cost function with respect to weights and bias is computed.

$$dW = \frac{\delta}{\delta W} C(W,b) \quad (13)$$

Where W, b is it weight and bias

Partial differentiation with respect to bias

$$db = \frac{\delta}{\delta b} C(W,b) \quad (14)$$

Goal is to Minimize $C(W,b)$

4. Result Analysis and Discussion

The SARSA algorithm learns a policy that balances exploration and exploitation and can be used in a variety of applications, including robotics, game playing, and decision making. However, it is important to note that the convergence of the SARSA algorithm can be slow, especially in large state spaces, and there are other reinforcement learning algorithms that may be more effective in certain situations. The proposed approach of

The weights and bias are then updated by making use of gradients of the cost function and learning rate α [0,1]. The value of α can range from 0.0 to 1.0. Greater the value of α, greater is the number of steps taken to find the global minimum of the cost function.

Updating parameters: W=W – (α . dW) and b=b-(α . b) (15)

MQLMS is implemented with multi agent environment and the agent is of type, deep Q agent value based.

In experiment an EC2 c5.9xlarge instance with 36 vCPUs is used. SparkSQL's bushy dynamic pro-gram takes 1000 seconds to plan the largest query (Q64, 18-relation join); A zoomed-in view of the rest of the planning latencies is included. Results in Fig1 the y = x line represent speedup. Across the workload, DQ's mean speedup over SparkSQL for execution is 1.0× and that for optimization is 3.6×. Below figure 2 showed the latency comparison.

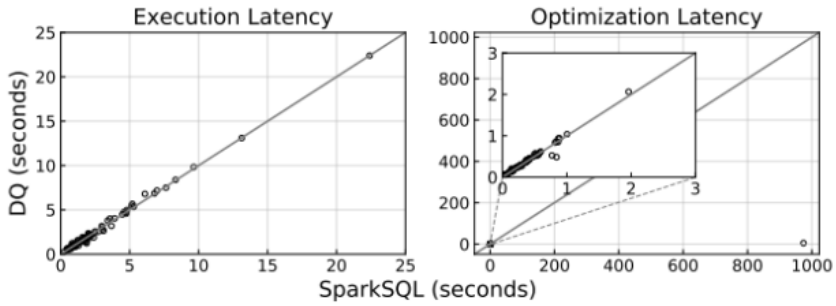

Fig 2: Execution and optimization latencies of DQ and SparkSQL on TPC-DS (SF1).

As shown in Fig 3 ,A modest amount of real execution using around 100 queries allows DQ to surpass both its original performance (by 3×) as well as Postgres (by 3.5×).

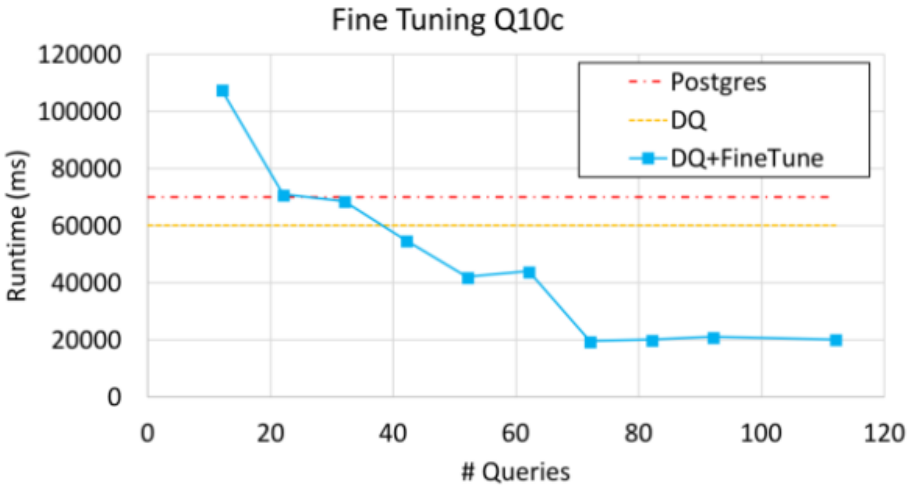

Fig 3: Effects of ne-tuning DQ on JOB Q10c.

As shown in Fig 4 (a) and (b) as the training steps increases the RL algorithm learns from environment and reduces the losses and improves accuracy. Here training steps are normalized between the range [0,1]. The accuracy which was targeted through training is obtained during validation when the training steps are increased by 50% whereas loss difference between training and validation reduces marginally after 80% steps.

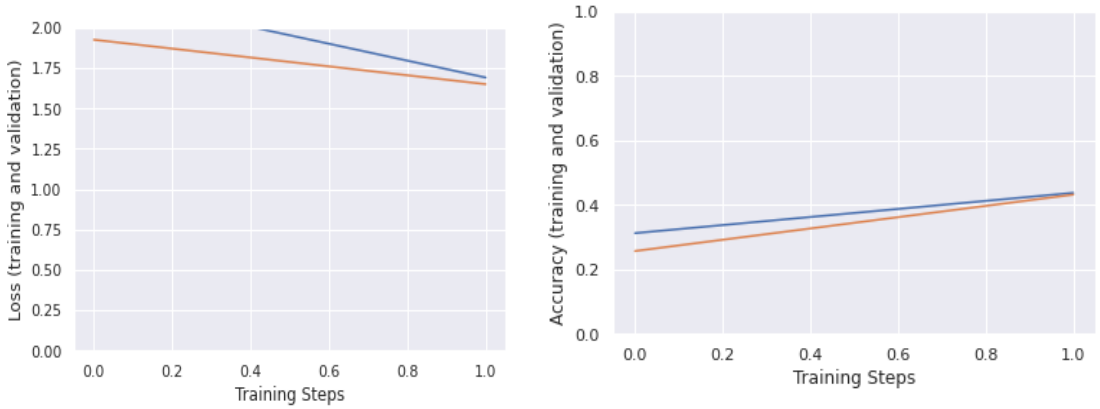

Fig 4(a): Loss during training and validation

Fig 4(b): Accuracy vs Training steps

Comparative Cost analysis for existing all three cost models and proposed cost model shown in below table1.

Table1: Comparative Cost analysis for existing cost models and proposed cost model

	Proposed Cost Model			Cost Model 1			Cost Model 2			Cost Model 3		
Optimizer Defined in various literature	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
Zig-zag (ZZ) [16]	1.02	5.96	49.56	1.0	1.07	1.87	1.0	5.07	43.16	1.0	3.41	23.13
QickPick (QP) [24]	1.01	65.08	498.34	1.0	23.87	405.04	7.43	51.84	416.18	1.43	16.74	211.13
IK-KBZ (KBZ) [25]	1.00	37.78	118.67	1.0	3.45	36.78	5.21	29.61	106.34	2.21	14.61	96.14
Right-deep (RD) [26]	1.56	98.04	387.63	4.7	53.25	683.35	1.93	8.21	89.15	1.83	5.25	69.15
Left-deep (LD) [26]	1.04	12.87	78.32	1.0	1.08	2.14	1.75	7.31	65.45	1.35	4.21	35.91
Exhaustive (EX) [26]	1.00	1.06	1.28	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Existing DQ	1.03	1.98	14.92	1.0	1.32	3.11	1.0	1.68	11.64	1.0	1.91	13.14
Modified DQ (Proposed)	1.09	2.03	18.92	1.0	1.78	4.12	1.0	1.87	17.23	1.0	1.94	18.91

Table 1 compares a proposed cost model with three parameters min, max and mean .It is observed that in proposed cost model large variation of cost with distinct clusters of minimum cost projected ,as mean value is very close to minimum value of cost. Modified DQ techniques projected better outcomes in comparison of existing DQ and other existing optimizers. The proposed cost model has minimum cost of 1.09 and range is larger as 18.92 with mean value of 2.03 which is close to minimum cost .

Although the modified Q cost-model shifts the mean cost less towards the lowest cost value, the broad cost range has grown by 28.36% compared to the original DQ. With regard to the change in memory size from 108 to 102, the

proposed cost-model's improved DQ results in an 88.2% reduction in variation for the relative mean cost.

Table 2: Proposed Cost Model mean relative cost vs. memory limit

	$M=10^8$	$M=10^6$	$M=10^4$	$M=10^2$	D= $M=10^8$- $M=10^2$
KBZ	1.0	2.96	28.03	30.78	29.78
LD	1.0	1.04	5.72	6.45	5.45
EX	1.0	1.0	1.01	1.02	0.02
Existing DQ	1.0	1.21	2.41	5.24	4.24
Modified DQ	1.02	1.34	1.42	1.52	0.5

Table 2 shows that proposed modified DQ algorithm utilized minimum resources for maximum learning and perform better than majority of the existing algorithms. Difference D is 0.5 in memory allocation from 102 to 108 whereas for KBZ optimizer it is 29.78.

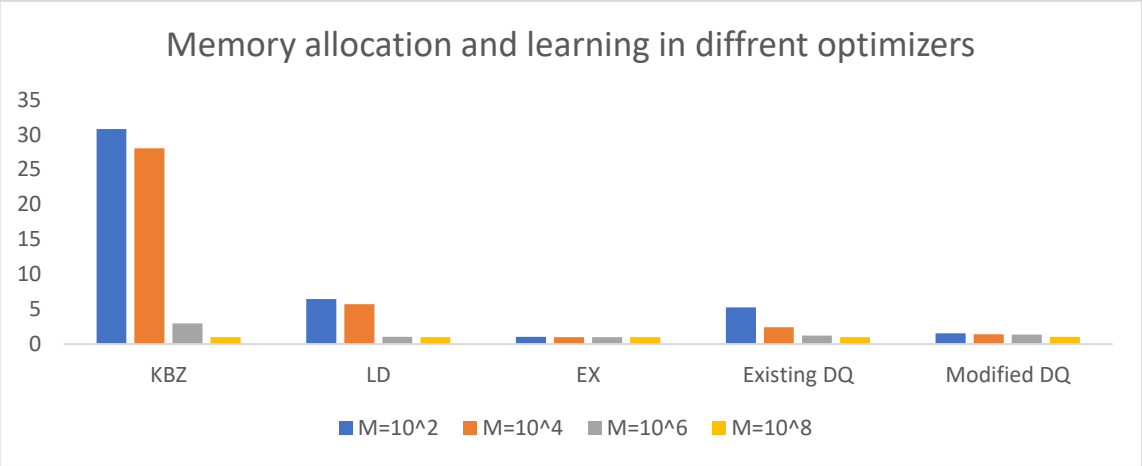

Fig 5: Memory allocation and learning in various optimizers

As shown in Fig 5 the difference in learning is almost flat which ensures that with minimum resources also optimum learning can be ensured. Table 3 shows proposed cost model training execution time vs. memory limit (number of tuples in memory). (Relative Scaling* in Milli-seconds) *960.02 mili-sec for 8 GB RAM and Internet speed approx. 100 MBPS is considered.

Table 3: shows Proposed Cost Model Training Execution time in ms vs. memory limit

	$M=10^2$	$M=10^4$	$M=10^6$	$M=10^8$	Difference
KBZ	50.23	10.33	5.2	1.04	49.19
LD	27.32	8.67	3.41	1.91	25.41
EX	2.5	1.03	0.96	0.92	1.58
Existing DQ	2.24	2.01	1.67	1.24	1
Modified DQ	1.84	1.56	1.2	1	0.84

Table 3 shows that as memory increases time reduces for execution. In Modified DQ algorithm the difference in time requirement compared to all other optimizers is minimum. Maximum time is taken by KBZ optimizer.

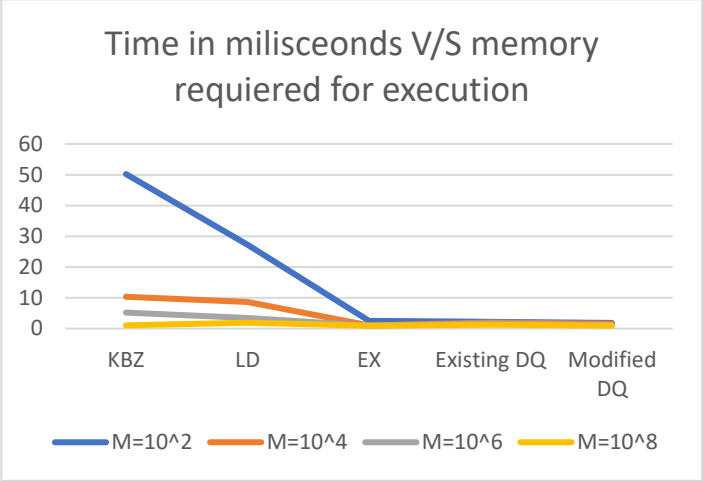

Fig 6: Time required for different memory allocation

As shown in Fig 6 it is clearly observed that modified DQ algorithm is efficiently execute the task with minimum memory. Below table 4 shows proposed cost model accuracy vs. memory limit.

Table 4: Proposed Cost Model Accuracy% vs. memory limit for specific training execution time (number of tuples in memory).

	$M=10^2$	$M=10^4$	$M=10^6$	$M=10^8$
KBZ	67	73	87	53
LD	78	81	85	64
EX	84	89	91	76
Existing DQ	83	91	94	78
Modified DQ	86	94	97	82

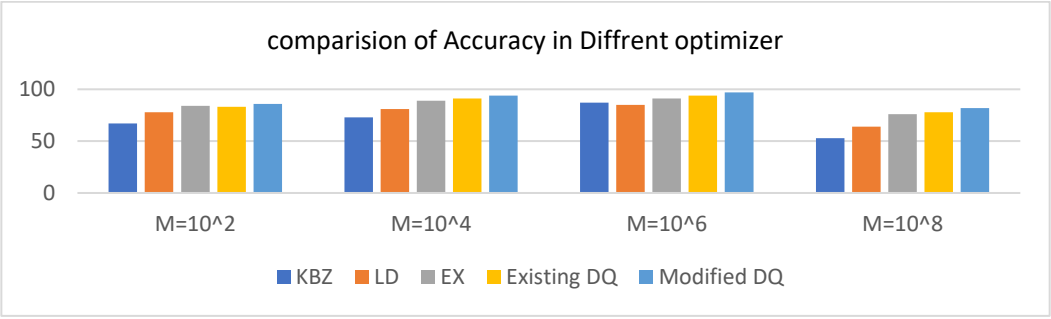

Fig 7: Accuracy comparison of different optimizers

As shown in Fig 7 the accuracy of Modified DQ optimizer is better in different memory allocations compared to existing optimizers

5. Conclusions

MDQL is a practical compromise that takes advantage of the join optimisation problem's structure. Proposed cost model gives wide range of weight parameters “W” where the mean value is always closer to the minimum cost which implies easy to cluster out and train the features. The modified DQ in proposed cost-model although has less shift in mean cost towards minimum cost value, there is a wide cost-range increased by 28.36% compared to existing DQ. Proposed modified DQ gives 88.2% reduction in variation for relative mean cost with respect to change in memory size from 10^8 to 10^2 for proposed .cost-model. For proposed cost model, training execution time for modified DQ is reduced by 19.35% for memory size of 10^8 and by 17.85% for memory size of 10^2 compared to existing DQ whereas accuracy has been

increased by 3 to 4%. Future research that focuses on the extremes of learning and query optimisation may reveal more information.

References

[1] Nguyen T T, Nguyen N D & Nahavandi S 2020 Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Trans Cybern*. 50(9):1-27

[2] Cao Z & Lin C T 2023 Reinforcement Learning From Hierarchical Critics. *IEEE Trans Neural Netw Learn Syst*. 34(2):1066-1073

[3] Li, X., Xu, H., Zhang, J., & Chang, H.2023 Deep Reinforcement Learning for Adaptive Learning Systems. *Journal of Educational and Behavioral Statistics*. 48(2): 220–243

[4] Tan, F., Yan, P., & Guan, X. 2017 Deep Reinforcement Learning: From Q-Learning to Deep Q-Learning. *International Conference on Neural Information Processing*.

[5] Park, J. & Park, J. 2020 Enhanced Machine Learning Algorithms: Deep Learning, Reinforcement learning and Q-learning. *Journal of Information Processing Systems*. 16(5):1001-1007

[6] Wang Hao-nan, Liu Ning, ZhangYi-yun, Feng Da-wei, Huang Feng, Li Dong-sheng et.al. 2020 Deep reinforcement learning: a survey. *Frontiers of Information Technology & Electronic Engineering.* 21

[7] Xu Z X, Cao L, Chen X L, Li C X, Zhang Y L, & Lai J 2018 Deep reinforcement learning with sarsa and Q-learning: A hybrid approach. *IEICE Transactions on Information and Systems E101*. D(9): 2315-2322

[8] Ren J, Ye C & Yang F 2021 Solving flow-shop scheduling problem with a reinforcement learning algorithm that generalizes the value function with neural network. *Alexandria Engineering Journal*. 60(3):2787-2800

[9] Samieiyeganeh M., Rahmat R W B O K, Khalid F B, & Kasmiran KA 2022 Deep reinforcement learning to multi-agent deep reinforcement learning. *J of Theoretical and Applied Information Technology*. 100(4):990-1003

[10] Cai Q,Cui C,Wang W,Xie Z,Zhang M 2023 A Survey on Deep Reinforcement Learning for Data Processing and Analytics. *IEEE Transactions on Knowledge & Data Engineering*, 35(05):4446-4465

[11] Duryea E, Ganger M, &Hu W 2016 Exploring Deep Reinforcement Learning with Multi Q-Learning. *Intelligent Control and Automation*.07(04):129-144

[12] Gerber RH 1986 Dataflow query processing using multiprocessor hash partition algorithms. *Technical report, Wisconsin Univ., Madison*

[13] Markl V, Raman V, Simmen D, Lohman G, Pirahesh H, & Cilimdzic M 2004 Robust query processing through progressive optimization.Proceedings of the 2004 *ACM sigmod international conference on management of data*.659–670

[14] Neumann T, & Radke B 2018 Adaptive optimization of very large join queries. In Proceedings of the 2018 *International Conference on Management of Data*. 677–692

[15] Ortiz J, Balazinska M, Gehrke J, & Keerthi SS 2018 Learning state representations for query optimization with deep reinforcement learning. *In Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning, DEEM'18*. 1–4

[16] Van Hasselt H, Guez A, & Silver D 2016 Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*.30(1)

[17] Wu W, Chi Y, Hac ́ıgum u ̈s H,& Naughton J F 2013 Towards predicting query execution time for concurrent and dynamic database workloads. *Proceedings of the VLDB Endowment*.6(10):925–936

[18] Wu W, Chi Y, Zhu S, Tatemura J, Hacigum ̈us H, & Naughton J F 2023 ̈Predicting query execution time: Are optimizer cost models really unusable?. *Proceedings International Conference on Data Engineering.* 1081-1092

[19] Leis V, Gubichev A, Mirchev A, Boncz P, Kemper A, & NeumannT 2015 How good are query optimizers, really? *Proceedings of the VLDB Endowment*. 9(3):204–215

[20] Uehara M, Shi C & Kallus N 2022 A Review of Off-Policy Evaluation in Reinforcement Learning.

[21] Zhao D , Wang H, Shao K, & Zhu Y 2016 Deep reinforcement learning with experience replay based on SARSA. *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. 1-6

[22] Bellman R E 1957 Dynamic programming. *Princeton University Press*

[23] Tan F, Yan P, Guan X 2017 Deep Reinforcement Learning: From Q-Learning to Deep Q-Learning

International Conference on Neural Information Processing.475-483

[24] Waas F, &Pellenko A 2000 Join order selection-good enough is easy. *In British National Conference on Databases.* 51–67

[25] Krishnamurthy R, Boral H, & Zaniolo C 1986 Optimization of nonrecursive queries. *In VLDB*. 86: 128–137

[26] Ziane M, Za M,& Borla-Salamet P 1993 Parallel query processing with zigzag trees. *The International Journal on Very Large Data Bases*. 2(3):277–302