Quest Journals
Journal of Software Engineering and Simulation
Volume 8 ~ Issue 6 (2022) pp: 43-47
ISSN(Online) :2321-3795 ISSN (Print):2321-3809
www.questjournals.org

Research Paper

A Model of an Adaptive Training System for Crude Oil Fractional Distillation Using Reinforcement Learning

Ebenibo Ineba Hannah [1]*, Anireh V.I.E [2], Taylor O.E. [3]
*[1]Department of Computer Science, Rivers State University, Nigeria
[2]Department of Computer Science, Rivers State University, Nigeria
[3]Department of Computer Science, Rivers State University, Nigeria
Corresponding Author: Ebenibo Ineba Hannah

Abstract: *The application of Reinforcement Learning to real-time, rapidly changing situations has produced unpredictable outcomes over time. This is due to the inherent nature of such environments, where there is often a lack of the vast number of datasets needed to guide learning agents, and even such data as can be found is frequently unstructured. A possible answer to these problems is the combination of RL with environment simulation. In this research, a solution to such issues has been proposed; M-QL. M-QL, a modification of the classic Reinforcement Learning Algorithm; Q-Learning, has been used as a technique for an Adaptive Training System that operates the console in an Oil and Gas Fractional Distillation Tower. The algorithm was designed to initialize the Q values of a learning agent with a Base policy that was trained using a small dataset obtained from the environment, which in turn was used to simulate various environmental situations to which the agent has to respond and control. Results showed that by using M-QL, the number of episodes required for training the agent, as well as running time were reduced by an average of 26% compared to Q-Learning. Varying the learning rate for both techniques also showed that this rate remained consistent, with M-QL maintaining an advantage over Q-Learning, even though the number of episodes and execution time decreased in both instances.*

Keywords*Reinforcement Learning, Scenario, Q-Learning, Reward Function, Learning Efficiency*

R*eceived 15 June, 2022;* ***R****evised 28 June, 2022;* ***A****ccepted 30 June, 2022 © The author(s) 2022.*
P*ublished with open access at* ***www.questjournals.org***

I. Introduction/Background

Reinforcement Learning is a type of Machine Learning Algorithm in which the learning agent is deployed to and exists in an environment which has different states. These states can be perceived by the agent and result in it taking various actions. These actions bring different rewards and also change the state of the environment. Thus, through a process of trial and error, the agent learns behaviours that lead to optimal actions and result in maximum rewards. [7].

Reinforcement Learning has been useful in various areas such as recommender systems, computer systems, energy, finance, healthcare, robotics, and transportation. [15]

Even though it is used in a number of systems, one of the challenges unique to Reinforcement Learning is determining the ideal balance between exploration and exploitation. The agent has to exploit what it has experienced within the environment in order to get a reward, but it also has to explore more of the environment in order to take better actions in the future. This requires planning, analysing the interaction between the plan and real-time environmental states and agent actions, as well as proper design of environmental models.[23] [25]. This makes Reinforcement Learning very suitable to be enhanced by the addition of Simulation.

Simulation is used to predict the behaviour of a system or process in a particular situation. In Computer Science, Simulations are equation-based programs that explore the approximate behaviour of a mathematical model. Typically, Simulation consists of two phases: Model Generation, where a deductive model is identified and Model Application, where the model is used to create simulation results. This is particularly useful for Reinforcement Learning because it provides additional training data in a controlled environment, as well as modelling real-time state changes in the environment. [20][29].

Q-Learning is a Reinforcement Learning Algorithm that assesses the quality of an action that is taken to move to a state, rather than the value of the state. It was developed by Chris Watkins at Cambridge University

in 1989. It is used in circumstances where there is insufficient knowledge of the dynamics of a real-world environment.

The 'Q' in Q-Learning stands for Quality. The quality being looked for in the algorithm is an evaluation of how useful an action is in aim of gaining a reward, as the agent moves from one state to another. The basic Q-Learning procedure comprises of initializing the Q Table with a set randomly chosen set of base values. The learning agent then choses and performs an action out of the possibilities presented, measures how much that action moved it towards the desired goal and then updates the Q Table. These steps are repeated until it reaches the required goal state or a condition for episode termination is met. [30]

A Q-Table is a data structure used to calculate the maximum rewards for each state, using the Q-function. It is made up of x number of columns, which represent the actions that can be taken and y number of rows, which represent all the possible states in an agent's environment. Each intersection between columns and rows represents the Q-value reward of taking a particular action at that particular state, denoted as Q(s,a).

The Q-function is the equation used to update the Q-Table. It is derived from the Markov Decision Process (MDP), The Bellman State value equation and the concept of Temporal Difference (TD). It contains both the current Q-Value, immediate past Q-Value, best Q-Value compared to all Q-Values derived thus far and the Reward function and states as well as the Learning Rate and Discount Factor.[9][17][28]

Q-Learning is widely used but lacks the ability to estimate values for unexplored states, this can lead to errors in approximation that can cause the agent to choose sub-optimal actions. [3].

II. Materials and Methods

In this work, developing the model required the study and use of the following factors.

- **Fractional Distillation Tower Concepts:**The Distillation Tower is the first and most important step in the crude refining process. Crude oil is basically made up of various hydrocarbon components that are mixed together. These components have different sizes, weights and boiling temperatures. The crude oil is heated and fed into the Distillation tower, and as the vapours progress up the tower, the components cool down after passing through trays, each at its own boiling point. At each tray, the separated component is siphoned out to a storage chamber or taken for further processing (condensing, cracking, unification or alteration). At the top of the tower, gas is either collected or flared. All these processes are usually handled by highly trained Operator Technicians in a control room. A typical Distillation tower is shown in Figure 1 below:

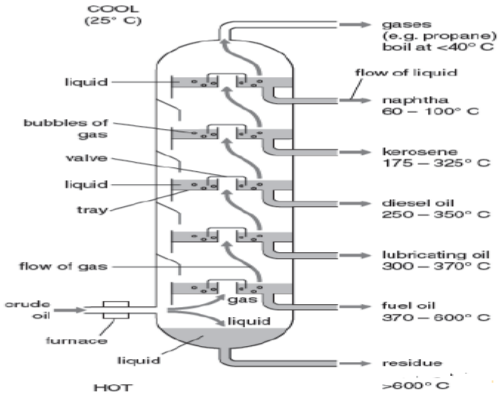


Figure 1: Fractional Distillation Tower

- **Input Specifications:**The proposed system is using simulated scenarios to evaluate its performance in handling the distillation process. The system requires data derived from real life oil refinery operations, as a basis for beginning the simulations, which can then be modified using GAN and Q-Learning.

The initial data can be obtained from operational equipment sensor readings within a Distillation Tower unit in a refinery. These sensors typically give readings on various conditions including:

- Feed Volume: This is the volume of crude flowing into the Distillation Unit over a period of time.
- Distillation Unit Temperature: This is the temperature within the Distillation Unit, ranging from hottest at its base, where the temperature control furnace resides, to coolest at the top, where gases are released (flaring)
- Distillation Unit Pressure: This is the measurement of pressure within the Distillation Unit, which is very important because too little pressure slows down the refining process, while too much results in gas flaring.

The input information for the proposed system will be restricted to these measurements, so as to limit the number of environmental states being modelled within the simulations.

- **Q-Value Specifications:**The State space will be represented with 3 values for each of the measurements of Volume, Temperature and Pressure which will designate of they are too little (0), Acceptable

(1) or too much (2), in a tuple Si = {V, T, P}. For example, S1 = {V1, T0, P2}.This will give a total of 84 States, describing the current environment state of the simulation at any time.The Action space will be defined with 2 values, describing an action to either reduce a state measurement (A0), or increase it (A1).The Reward Function will be defined as R{V,T,P} where the score for the complete set of states at any given time will be as follows:
R{1,1,1} = +15
R{0,0,0} = -10
R{2,2,2} = -10
R{Every double '1' State} = +5
R{Every double '0' State} = -2
R{Every double '2' State} = -2
R{Every other State not included in 1 – 6 above} = 0
Likewise, the conditions for episode termination will be given as:
100L Volume distilled.
Reward Score = -50 or less.

- **System Architecture:**

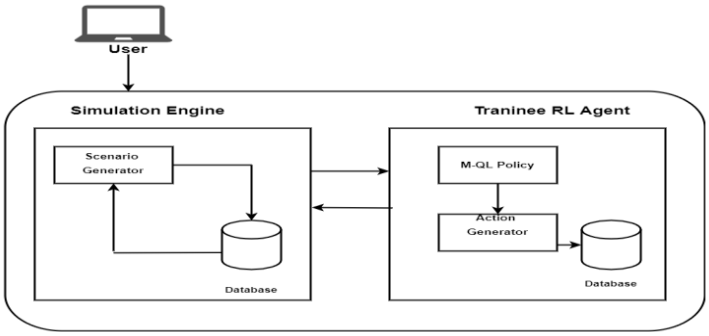


Figure 2: System Architecture

- **M-QL:**

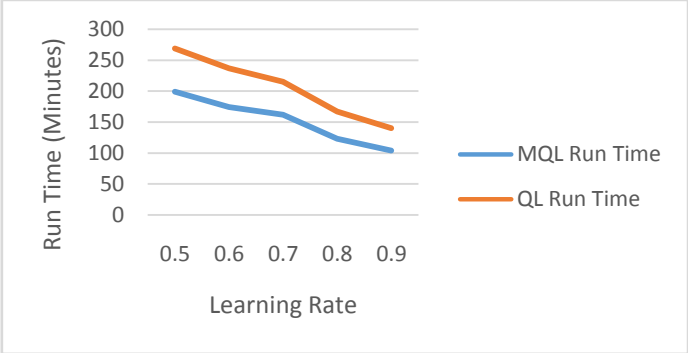
In order to make use of the inherent information gathered by simulation, a modified version of Q-Learning, called M-QL was used for the Q-Value update. The M-QL Algorithm is given below:
1: Input initial data set D (s,a,r)
2: Extract Base Policy P0(s,a,s′) from D
3: Use P0 to initialize Q(s, a)
4: for each episode do
5: Initialize s
6: for each step of episode (until s is terminal) do
7: Choose a from s using policy derived from Q (e.q. ϵ-greedy)
8: Take action a, observe r, s′
9: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)]$
10: s ← s′
11: end for
12: end for

III. Results & Discussion

Table 1:Q-L and M-QL Learning Efficiency Comparison

ALGORITHM	MQL	MQL	QL	QL
Learning Rate	No. of Episodes until Optimal Policy	Run Time (minutes)	No. of Episodes until Optimal Policy	Run Time (minutes)
0.5	2362	199	3232	269
0.6	2084	174	2844	237
0.7	1941	162	2591	215
0.8	1475	123	2015	167
0.9	1250	104	1680	140

Figure3: Graph Comparison of Learning Efficiency of M-QL with Q-Learning



IV. Conclusion

Q Learning uses significant time and computational resources for appropriate exploration and has difficulty producing consistent outcomes when used in environments that change at a rapid pace. This study aimed to show that effects of the application of Simulations scenarios based on datasets applied to Q-Learning to change it into M-QL. This has led to an increase in performance as shown by such indicator as the reduction of run time and increase in Learning efficiency. Further enhancement by using two learning agents within a training scenario, with each agent sharing experiences should be explored.

Acknowledgment

I would also like to thank my supervisor, Dr. V.I.E. Anireh, for his patience, guidance and contributions to the work.

References

[1]. Aimsun T. (2012). Dynamic Simulators User's Manual. Transport Simulation Systems, 20, 2012.
[2]. Beakcheol J., Myeonghwi K., Gaspard H., & Jong Wook K. (2017). Q-learning Algorithms: A Comprehensive Classification and Applications. IEEE Access. 20 (8). 2941229
[3]. Busoniu L, Babuska R, & De Schutter B. (2016). A Comprehensive Survey of Multiagent Reinforcement Learning. IEEE Transactions on Systems, Man, And Machine.
[4]. Everitt T., Hutter M., Kumar R., &Krakovna V. (2021). Reward Tampering problems and Solutions in Reinforcement Learning. Australian National University.
[5]. Francois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., &Pineau, J. (2018). An Introduction to Deep Reinforcement Learning. Foundations and Trends® in Machine Learning, 11(4), 219–354.
[6]. Gourav K. & Kaur A. (2020). A Study of Reinforcement Learning Applications and its Algorithms. In International Journal of Scientific & Technology Research, 9(3). 4223-4228
[7]. Glatt R. &Reali Costa A.H. (2017). Improving Deep Reinforcement Learning with Knowledge Transfer. In Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17). Association for the Advancement of Artificial Intelligence. 17 (5036-5037)
[8]. Hu J. & Wellman M.P. (2003). Nash Q Learning for General Sum Stochastic Games, Journal of Machine Learning, 4 (1039 1069, 2003).
[9]. Jang B, Gaspard H., Jong Wook K., &Myeonghwi K. (2017). A Comprehensive Classification of RL Algorithms and Applications. The Institute of Electrical and Electronics Engineers. 20(17).
[10]. Jiang, T., Gradus, J. L., &Rosellini, A. J. (2020). Supervised Machine Learning: A Brief Primer. Behavior Therapy, 51(5), 675–687.
[11]. Kaelbling, L., Littman, M., & Moore, A. (1996). Reinforcement learning: a survey. Journal of Artificial Intelligence Research, 4, 237–285.
[12]. Kurenkov A. (2018). Reinforcement Learning's Foundational Flaw. The Gradient. July 8th, 2018
[13]. Law AM, Kelton WD. (2015) Simulation Modeling and Analysis. New York: McGraw-Hill; 5th Edition. (85-102)
[14]. Li Y. (2019). Reinforcement Learning Applications. Cornell University. arXiv:1908.06973v1 – arXiv.org
[15]. Lonza A. (2019) Reinforcement Learning Algorithms with Python. Packt Publishing. 1st Edition (74-134)
[16]. Lu Shoufeng, Liu Ximin, & Dai Shiqiang. (2008). Q-learning for adaptive traffic signal control based on delay minimization strategy. Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference, (687–691).
[17]. Moerland M.T., Broekens J., &Jonker M. C., (2021). Model-Based Reinforcement Learning: A Survey. Leiden University, The Netherlands. arXiv:2006.16712v3 – arXiv.org
[18]. Nkuma-Udah K.I. (2020). CIT 902: ICT Research Methodology and Statistics. National Open University of Nigeria. (30-67)
[19]. Paternina-Arboleda C., Montoya-Torres J., & Fabregas-Ariza A. (2008). Simulation Optimization Using a Reinforcement Learning Approach. 2008 Winter Simulation Conference.
[20]. Saad E. (2010). Bridging the Gap between Reinforcement Learning and Knowledge Representation: A Logical Off- and On-Policy Framework. 11th European conference on Symbolic and quantitative approaches to reasoning with uncertainty.
[21]. Shiue, R., Lee K., & Su C. (2018) Real-time scheduling for a smart factory using a reinforcement learning approach. Computers & Industrial Engineering; 125(604-614)
[22]. Shlaer S. & Mellor S. (1992) Object Life Cycles: Modelling the World in States, Yourdon Press, Englewood Cliffs, NJ
[23]. Stricker N, Kuhnle A, Sturm R &Friess S. Reinforcement learning for adaptive order dispatching in the semiconductor industry. CIRP Annals 2018;67:511-514.
[24]. Sutton R.S. &Barto A.G. (2018). Introduction to Reinforcement Learning. MIT Press, 2nd Edition. 1-13
[25]. Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for
[26]. temporal abstraction in reinforcement learning. Artificial Intelligence, 112(1-2):181–211.

[27]. Van Hasselt, H. (2010). Double Q-Learning. 24th Annual Conference on Neural Information Processing Systems, 2010
[28]. von Rueden L, Mayer S., Sifa R., Baukhage C., &Garcke J. (2020). Combining Machine Learning and Simulation to a Hybrid Modelling Approach: Current and Future Directions. International Symposium on Intelligent Data Analysis (IDA)
[29]. Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. Machine Learning, 8(3), 279–292.