International Journal of Big Data and Analytics in Healthcare
Volume 6 • Issue 2

Improvement in Task Scheduling Capabilities for SaaS Cloud Deployments Using Intelligent Schedulers

Supriya Sawwashere, Kalinga University, India

ABSTRACT

Task scheduling on the cloud involves processing a large set of variables from both the task side and the scheduling machine side. This processing often results in a computational model that produces efficient task-to-machine maps. The efficiency of such models is decided based on various parameters like computational complexity, mean waiting time for the task, effectiveness to utilize the machines, etc. In this paper, a novel Q-Dynamic and Integrated Resource Scheduling (DAIRS-Q) algorithm is proposed which combines the effectiveness of DAIRS with Q-Learning in order to reduce the task waiting time and improve the machine utilization efficiency. The DAIRS algorithm produces an initial task-to-machine mapping, which is optimized with the help of a reward and penalty model using Q-Learning, and a final task-machine map is obtained. The performance of the proposed algorithm showcases a 15% reduction in task waiting time and a 20% improvement in machine utilization when compared to DAIRS and other standard task-scheduling algorithms.

KEYWORDS

Cloud Deployments, Cloud-Based IoT, DAIRS, Intelligent Schedulers, Machine Mapping, Machine Utilization, Q-Learning, Task Scheduling, Task Waiting Time

1. INTRODUCTION

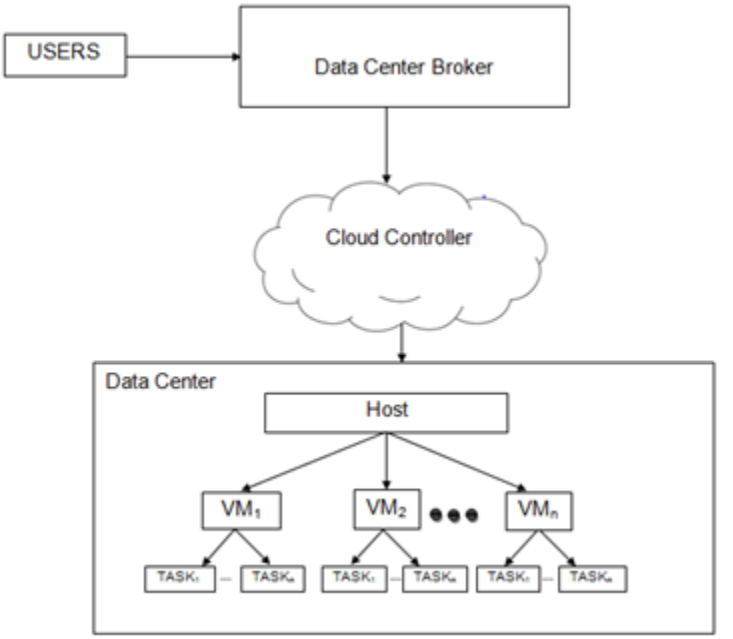
Scheduling tasks over the cloud is a multi-domain problem, which includes pattern analysis, filtering, classification, clustering and prediction. Usually the following processes are followed in order to schedule cloud tasks (Asghari et al., 2020),

- Identification of undertaking boundaries from the task dataset
- Identification of machine parameters from the asset pool
- Strategizing rules and thresholds for machine and task scheduling
- Task execution on the given machine
- Evaluation of irregularities in execution, and changing methodology based to output parameters
- Post processing of tasks and machines if needed

In view of these steps, the researchers can see that at first the task parameters must be investigated. These parameters must incorporate essential assignment measurements like the undertaking execution delay, the task cutoff time, the task holding up time, while they can likewise incorporate optional

DOI: 10.4018/IJBDAH.287104

Figure 1. A typical task scheduler



parameters like undertaking mutual exclusiveness, shared reliance, and others (Nawrocki & Sniezynski, 2020). Typically, the all-out assignment execution prerequisites are administered by condition 1,

$$T_{TE} = F\left(T_{ed}, T_{dl}, T_{wt}, [T_{sec}]\right)\ldots(1)$$

Where, T_{TE} is the total task execution requirement, T_{ed} is the total task execution delay, T_{dl} is the task deadline, T_{wt} is the task waiting time, while, T_{sec} are secondary application specific parameters needed to execute the task.

Once the task parameters are identified, then the resource parameters are observed, and evaluated. These parameters are again divided into primary and secondary parameters. Primary parameters include but are not limited to number of execution units available, capacity of each unit to execute the task, execution requirements for the resources, and others (Sui et al., 2019). A typical task scheduler can be observed from figure 1, wherein the tasks coming from users are given to the data center broker, the broker sends these tasks to the cloud controller for processing. The controller finally gives it to the host for further processing and scheduling on different machines.

The next section describes about such task scheduling systems in brief, and is followed by the proposed DAIRS-Q algorithms. This text further evaluates the said algorithm on different application specific datasets, and compares its efficiency with some state-of-the-art methods. Finally, the concludes with some interesting observations about the proposed protocol, and recommendations on how to further explore the field of work.

2. LITERATURE REVIEW

AI has become a true norm for task scheduling research. The work (Ge & Liu, 2020) uses Q-Learning for scheduling undertakings on a cloud-based Internet of Things (IoT) climate. The principle bit of leeway of Q-Learning is that, it gives a motivating force (reward) and punishment procedure while

finding out about the basic issue. For example, in Ge & Liu, (2020), analysts have utilized a worldwide view strategy for task scheduling, wherein the calculation's choice is compensated if the general cloud execution improved by it, or it is given a punishment if the cloud execution debases. The prize and punishment is as a mathematical worth, which is augmented by the calculation. They have isolated the errand nodes into energy unwinding and energy tense nodes. The energy tense nodes are answerable for imparting back and forth between the cloud and the clients. While the energy unwinding node assumes control over a portion of the heap structure the energy tense nodes. The principle point of the calculation is to improve the network lifetime, by enhancing the heap on every one of the node. A similar calculation can be applied to any sort of assignment scheduling applications, and it is prescribed that perusers apply it to assess its exhibition on various applications. The calculation can be stretched out by supplanting Q-Learning with fortification realizing, which is an unrivaled method for quicker union and better learning results. This can be seen from the work in Melnik & Nasonov, (2019), wherein fortification learning is joined with neural networks for scheduling work processes. The engineering takes in the assignments which need scheduling, and offers them to a processing climate. The figuring climate assesses a prize capacity dependent on the scheduling done by the neural network. Normally the neural network plans errands arbitrarily, and attempts to limit the blunder in scheduling with the assistance of learning models like Levenberg–Marquardt, inclination plummet, and so forth Support learning can be broadened utilizing a more intricate preparing condition set, which can think about both essential and auxiliary boundaries for undertakings and execution units the same. For example, the work in Waschneck et al., (2018), broadens support learning, and assesses a profound fortification learning component dependent on Deep Mind which is Google's Deep Q-Learning Network (DQN). It utilizes a mix of Markov learning measure with directed learning (so the calculation find out about the execution climate), for better undertaking scheduling effectiveness. The model can improve the asset use, and decrease the undertaking holding up time, yet has a high displaying multifaceted nature. Since each time something changes in the processing plant climate, a comparative change must be made in the computerized twin climate. Generally, ongoing undertaking execution frameworks have an enormous number of changes happening occasionally, subsequently this system isn't appropriate for such exceptionally powerful conditions. However, the exploration accomplished for planning a 2-level learning calculation is excellent, and hence is utilized as the reason for this fundamental examination.

Another Q-Learning based calculation is referenced in Kim et al., (2019), wherein an IoT network comprising of temperature, mugginess and weight sensors is thought of. Here, a MDP or Markov choice cycle is applied, which totals the prizes given to every arrangement set to locate the last prize worth. The work in Waschneck et al., (2018) and Kim et al., (2019) is stretched out in Zhang et al., (2019), wherein a profoundly dispersed climate is considered for task scheduling in programming as a help (SaaS) cloud. They have utilized the idea of versatile unique programming (ADP) to plan undertakings dependent on both assignment and execution unit boundaries. It tends to be seen that the proposed component performs in a way that is better than a portion of the cutting-edge calculations, and can be utilized as a decent sending methodology when scheduling assignments in a conveyed climate.

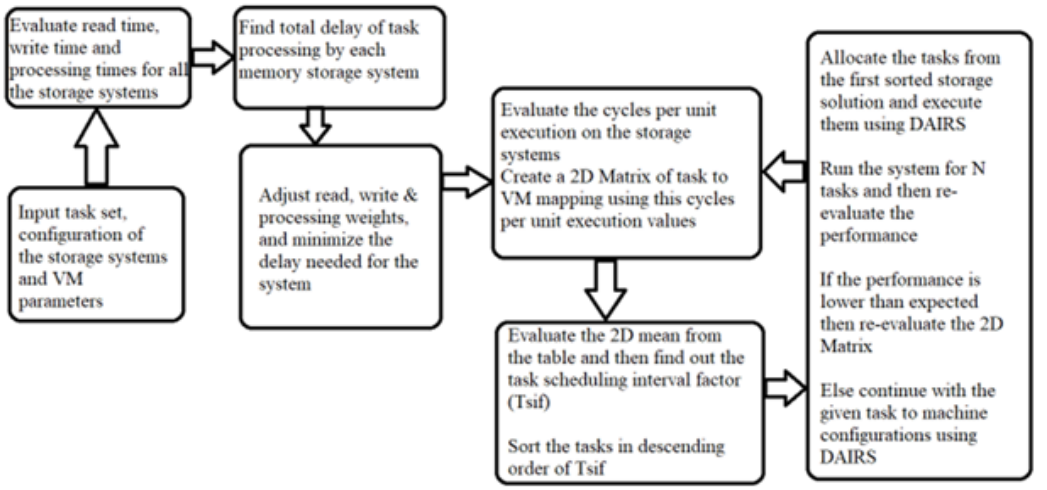
A use of this calculation can be seen from Hu et al., (2019), wherein the errand scheduling issue is first changed over into min-max number direct programming (MMILP), and afterward it is changed over to a distinguishable curved target work with a unimodular imperative grid for adding non-linearity. The capacity and the framework information gain from the conveyed disseminated network, and improve the undertaking scheduling productivity by diminishing the computational deferral, and lessening the quantity of bytes moved during scheduling. The work in Nawrocki et al., (2020) presents an energy viable arrangement, uses energy relevant boundaries like CPU use, network association, defer required for calculations, and so on to assess the best scheduling procedure. A similar setting mindfulness can be utilized to upgrade some other boundary, similar to defer required for scheduling, mean holding up time, cutoff time hit proportion, and so forth Because of which this work has been chosen to advance certain boundaries in the fundamental exploration. The

energy utilization is diminished with the assistance of neighborhood learning, and the conveyed task scheduler can plan the assignments on the nodes that are the most energy proficient. This decreases the framework's productivity to enhance other undertaking and node boundaries, yet improves the general framework lifetime. Such a framework can be utilized where the undertaking to node scheduling needs to streamline just a single boundary. In any case, as number of boundaries increment, the setting of the issue will in general change, and complex calculations like Stavrinides & Karatza, (2017), that utilizations rough calculations for task-scheduling can be used. Because of this, the settle on range and administration level understanding infringement proportion lessens, accordingly improving the errand scheduling proficiency.

Another CNN roused cross breed profound neural network scheduler is talked about in Zang et al., (2019), which uses convolution two-dimensional change technique to perform task scheduling. Another 2-stage task scheduler is portrayed in Zhang & Zhou, (2017), which uses both current and recorded information for scheduling errands. Because of the utilization of both authentic and current errand and node information, the calculation can foresee task examples and VM utility examples. A utilization of this framework can be seen in Zheng et al., (2020), wherein task scheduling is applied to savvy city situations. They presume that Extended Hungarian calculation with round support line (EHGC) is a superior calculation when contrasted and FIFO and other shortsighted frameworks. The use of profound learning and fortification learning can be tried for savvy urban communities. Scheduling can likewise be conveyed as a cloud administration, the work in Moorthy & Pabitha, (2019) features a utilization of sending scheduling calculations on the cloud, and giving it as assistance. It tends to be utilized as a strong use-case for planning scheduling calculations.

An epic particle swarm optimization (PSO) calculation for task scheduling is depicted in Ebadifard & Babamir, (2017), wherein the wellness work is changed to be a mix of make-range and the asset use. A Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) calculation is applied in Khorsand & Ramezanpour, (2020), which uses the best-most exceedingly terrible strategy for task scheduling. It additionally utilizes the oversimplified approach as utilized in Dong et al., (2019), and can accomplish comparative execution like Dong et al., (2019), but on the other hand can decrease the energy utilization because of the utilization of energy as a wellness boundary. However, both Dong et al., (2019) and Prasanna Kumar & Kousalya, (2019) have restricted use cases because of the restricted capacity of the calculation demonstrated, which can be improved by the work in Shobha Rani & Pounambal, (2019). Here, another profound fortification learning calculation is depicted that uses numerous errand and node boundaries for better optimization. Like PSO (Dong et al., 2019), a crow inquiry optimization (CSO) is portrayed in Zhou et al., (2018). It utilizes comparable exploration boundaries like Dong et al., (2019), and accomplishes a comparative execution on shortsighted datasets. The work can be streamlined utilizing a profound learning network as depicted in Peng et al., (2019), wherein task offloading is improved preparing. It utilizes profound learning CNN for task scheduling, and gives better make range and better computational usage when contrasted and Zhou et al., (2018). A comparative way to deal with Zhou et al., (2018) is referenced in Huang et al., (2019), where an adjusted hereditary calculation (GA) joined with voracious methodology (MGGS) is applied. The GA utilizes a voracious methodology for wellness assessment and traverse activities, which makes it viable for scheduling undertakings. However, like Dong et al., (2019) and Zhou et al., (2018), the methodology has characteristic downsides because of set number of boundary use for wellness assessment. Accordingly, the work in Mostafavi & Hakami, (2020), which utilizes a profound Q-Learning network can be utilized for execution upgrade. Comparative profound learning techniques are referenced in Huang et al., (2019), Mostafavi & Hakami, (2020) and Tong et al., (2019), where Q-Learning and fortification learning is either utilized independently or joined with neural networks to accomplish unrivaled execution. Consequently, the underlying research likewise utilizes Q-Learning in mix with DAIRS to improve the framework execution for load adjusting over the cloud.

Figure 2. Proposed DAIRS-Q method for real time clouds



3. PROPOSED DAIRS-Q ALGORITHM

The original DAIRS algorithm (Anjum et al., 2020) does not perform well under hybrid storage conditions, and nowadays all cloud systems are based on hybrid storage (which combine SSDs, HDDs, etc.). Therefore, the proposed Q-learning based DAIRS is proposed that uses dummy reads and writes in order to evaluate the performance of the original algorithm, and get the final optimized scheduling results. The overall architecture of the proposed DAIRS can be observed from figure 2. The proposed hybrid DAIRS (DAIRS-Q) system, works in the following steps,

1. For a cloud with 'N' types of storage systems, arrange each of these systems in descending order of processing. Due to this the highest performance system is placed at the top, while the lowest performing system is placed at the bottom.
2. Perform 'N' dummy reads and 'N' dummy writes on each of the cloud systems, and observe the following parameters,
   a. The reading delay for all systems Tr1, Tr2, …, TrN
   b. The writing delay for all systems Tw1, Tw2, …, TwN
   c. The processing delay for all systems Tp1, Tp2, …, TpN

3. Evaluate the total time needed for processing one task using the following equation 1,

$$Td_i = w_r * Tr_i + w_w * Tw_i + w_p * Tp_i \quad (1)$$

where, w_r, w_w and w_p are the weights for reading, writing and processing. Higher values of weights indicate that the application needs that particular element to be enhanced. For instance, an application that requires faster processing will have higher value of w_p, while the values of w_r and w_w would be lower than that of w_p.

Table 1. Relationship between system memory type and input task

Tn1 *T1*	*Tn2* *T1*	*Tn3* *T1*	….	*TnN* *T1*
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Tn1 Tk	Tn2 Tk	Tn3 Tk	…	TnK Tk

4. Evaluate the lowest total completion time for each of these machines and store them into an array. Let the elements in this array be named as D1, D2, D3, … DN
5. For each storage system, the time needed to process the task on the system will be different. Let Tni represent the time needed for a task to be executed on that system, this can be evaluated with the help of equation 2 as follows,

$$Tni = \frac{Di*Ni}{Tdi} \tag{2}$$

where Tdi is the sum of processing time for the cloud, Ni is the total number of task units to be executed on the cloud, Di is the per unit task delay for the system.

6. Now evaluate the following dependency table, which indicates the execution time for a given task on a given set of machines,

7. This table consists of all 'k' tasks on the rows, and all the 'N' machine configurations on the column side

8. Find the double average value of the task execution threshold using the following equation 3,

$$Sth = \frac{\sum_{i=1}^{N}\sum_{j=1}^{k} Tn_i T_j}{N*k} \tag{3}$$

9. Now find the interval factor for the task using the following formula,

$$IF_t = \frac{Sth}{N} \tag{4}$$

Here, 'N' is used as the number of memory configurations are 'N' in the system.

10. Find the column-wise average value from the dependency table, and let the values of these averages be AT1, AT2, …, ATk
11. Arrange these tasks in descending order of the AT values, and let the new order for these tasks be Ts1, Ts2, ..., Tsk
12. For allocating a task 'i' to any machine, it must fulfill the given condition,

$$(i-1)*IF_t \leq ATd \leq i*IF_t \tag{5}$$

Where, 'd' is the ID of the task

Based on the equation 5, the task is given a penalty value if it doesn't follow equation 5, while it is given a reward value if the task follows it. All tasks are allocated to a machine only if they are rewarded, else the task is left unallocated. The process is repeated unless all tasks are successfully allocated by the system. Based on this allocation, the task with higher requirement is given to a memory system with better performance, while a task with lower requirement is given to a memory system with lower configuration/performance. This enables the task scheduler to schedule tasks with higher efficiency, and get better quality of experience performance for the end user. This performance can be observed from the next section, where a statistical comparison is made between the existing DAIRS and IDE algorithms with the proposed DAIRS-Q algorithm.

4. RESULT EVALUATION AND COMPARISON

To compare the performance of the proposed DAIRS-Q algorithm with the existing IDE (Wu et al., 2018) and DAIRS algorithms, the NASA load balancing dataset from the following website is used,

https://www.cs.huji.ac.il/labs/parallel/workload/l_nasa_ipsc/index.html

This dataset consists of more than 100k records with an average makespan of 600 ms for each task. Makespan is the average delay needed for execution of the task. Based on this dataset, the average cloud utilization ratio (CUR), the delay needed for task scheduling (Da) and the mean task waiting time (Tmwt) are evaluated. The following tables indicate the comparison of these values for different algorithms.

Similarly, the other parameters can be observed from table 2 and 3 as follows:

It can be observed that the proposed DAIRS-Q reduces the delay of task scheduling by 15% when compared with the existing DAIRS and IDE algorithms, while the cloud utilization ratio is improved by more than 10%. This happens due to the dummy reads and writes architecture which is proposed, that allows the system to evaluate the performance of the system before deployment, and therefore assigns the best task scheduling plan in place for the given set of tasks and virtual machine combinations.

Table 2. Comparison of cloud utilization ratio for different algorithms

VM-s	Tasks	CUR (DAIRS)	CUR (IDE)	CUR (DAIRS-Q)
10	1000	74.50	70.95	80.81
10	2000	75.60	72.00	82.00
10	5000	75.90	72.29	82.33
10	10000	76.30	72.67	82.76
10	20000	77.50	73.81	84.06
10	50000	79.10	75.33	85.80
10	100000	79.90	76.10	86.66
20	1000	80.51	76.68	87.33
20	2000	81.40	77.52	88.29
20	5000	82.29	78.37	89.25
20	10000	83.17	79.21	90.21
20	20000	84.06	80.05	91.17
20	50000	84.94	80.90	92.13
20	100000	85.83	81.74	93.09
50	1000	86.71	82.59	94.06
50	2000	87.60	83.43	95.02
50	5000	88.49	84.27	95.98
50	10000	89.37	85.12	96.94
50	20000	90.26	85.96	97.90
50	50000	91.14	86.80	98.86
50	100000	92.03	87.65	99.82

Table 3. Comparison of task execution delay

VM-S	Tasks	Da (s) (DAIRS)	Da (s) (IDE)	Da (s) (DAIRS-Q)
10	1000	5.20	4.95	4.13
10	2000	5.50	5.24	4.37
10	5000	5.70	5.43	4.53
10	10000	5.90	5.62	4.68
10	20000	6.23	5.93	4.94
10	50000	7.10	6.76	5.63
10	100000	7.50	7.14	5.95
20	1000	2.60	2.48	2.07
20	2000	2.75	2.62	2.18
20	5000	2.85	2.71	2.26
20	10000	2.95	2.81	2.34
20	20000	3.12	2.97	2.48
20	50000	3.55	3.38	2.82
20	100000	3.75	3.57	2.98
50	1000	1.30	1.24	1.03
50	2000	1.38	1.31	1.09
50	5000	1.43	1.36	1.13
50	10000	1.48	1.40	1.17
50	20000	1.56	1.48	1.23
50	50000	1.78	1.69	1.41
50	100000	1.88	1.79	1.49

Table 4. Comparison of mean waiting time for tasks

VM-S	Tasks	Tmwt (s) DAIRS)	Tmwt (s) (IDE)	Tmwt (s) (DAIRS-Q)
10	1000	6.90	6.57	6.12
10	2000	7.50	7.14	6.66
10	5000	8.90	8.48	7.90
10	10000	13.50	12.86	11.98
10	20000	14.90	14.19	13.22
10	50000	15.60	14.86	13.84
10	100000	16.20	15.43	14.38
20	1000	3.45	3.29	3.06
20	2000	3.75	3.57	3.33
20	5000	4.45	4.24	3.95
20	10000	6.75	6.43	5.99
20	20000	7.45	7.10	6.61
20	50000	7.80	7.43	6.92
20	100000	8.10	7.71	7.19
50	1000	1.73	1.64	1.53
50	2000	1.88	1.79	1.66
50	5000	2.23	2.12	1.97
50	10000	3.38	3.21	3.00
50	20000	3.73	3.55	3.31
50	50000	3.90	3.71	3.46
50	100000	4.05	3.86	3.59

5. CONCLUSION AND FUTURE SCOPE

Based on the result evaluation it can be observed that the proposed algorithm outperforms both DAIRS and IDE algorithms in terms of cloud utilization ratio, delay of task execution and mean task waiting time. The cloud utilization is improved by almost 10%, while the delay for execution is reduced by 15, and the mean waiting delay is reduced by 10% as well. This indicates that the proposed algorithm can be applied to any real-time cloud deployments, thereby improving the overall applicability of the proposed DAIRS-Q system. The proposed algorithm can be further improved with the addition of more sophisticated scheduling algorithms that do not require dummy reads and writes, because using these dummy requests increases the computational overheads on the system. In order to reduce them, machine learning techniques like deep-reinforcement learning can be used, along with better mapping algorithms for scheduling tasks on real-time clouds.

REFERENCES

Anjum, Chaudhary, & Karanjekar. (2020). Dynamic Load Balancing Scheduling Algorithm for Cloud Data Centers. *International Research Journal of Modernization in Engineering Technology and Science*, 1252-1256.

Asghari, A., Sohrabi, M. K., & Yaghmaee, F. (2020). Task scheduling, resource provisioning, and load balancing on scientific workflows using parallel SARSA reinforcement learning agents and genetic algorithm. *The Journal of Supercomputing*, 1–29.

Dong, T., Xue, F., Xiao, C., & Li, J. (2019). Task scheduling based on deep reinforcement learning in a cloud manufacturing environment. Wiley Online Library.

Ebadifard, F., & Babamir, S. M. (2017). A PSO-based task scheduling algorithm improved using a load balancing technique for the cloud computing environment. Wiley Online Library.

Ge, J., & Liu, B. (2020). Q-learning based flexible task scheduling in a global view for the Internet of Things. Wiley Online Library.

Hu, Li, & Luo. (2019). Time- and Cost- Efficient Task Scheduling across Geo-Distributed Data Centers. *IEEE Transactions on Parallel and Distributed Systems*, 705-718.

2Huang, J., Li, S., & Chen, Y. (2019). Revenue-optimal task scheduling and resource management for IoT batch jobs in mobile edge computing. *Peer-to-Peer Networking and Applications*, 1–12.

Khorsand, R., & Ramezanpour, M. (2020). An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing. Wiley Online Library.

Kim, D., Lee, T., Kim, S., Lee, B., & Youn, H. Y. (2019). Adaptive packet scheduling in IoT environment based on Q-learning. *Journal of Ambient Intelligence and Humanized Computing*, 1–11. doi:10.1007/s12652-019-01351-w

Melnik, M., & Nasonov, D. (2019). Workflow scheduling using Neural Networks and Reinforcement Learning. *8th International Young Scientist Conference on Computational Science*, 29-36. doi:10.1016/j.procs.2019.08.126

Moorthy & Pabitha. (2019). Optimal provisioning and scheduling of analytics as a service in cloud computing. Wiley Online Library.

Mostafavi, S., & Hakami, V. (2020). A Stochastic Approximation Approach for Foresighted Task Scheduling in Cloud Computing. *Wireless Personal Communications*, *114*(1), 1–25. doi:10.1007/s11277-020-07398-9

Nawrocki, P., & Sniezynski, B. (2020). Adaptive Context-Aware Energy Optimization for Services on Mobile Devices with Use of Machine Learning. *Wireless Personal Communications*, 1–29.

Nawrocki, P., Sniezynski, B., Kolodziej, J., & Szynkiewicz, P. (2020). Adaptive context-aware energy optimization for services on mobile devices with use of machine learning considering security aspects. *20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, 708-717. doi:10.1109/CCGrid49817.2020.00-19

Peng, Lin, Cui, Li, & He. (2019). A multi-objective trade-off framework for cloud resource scheduling based on the Deep Q-network algorithm. *Cluster Computing,* 1-15.

Prasanna Kumar, K. R., & Kousalya, K. (2019). Amelioration of task scheduling in cloud computing using crow search Algorithm. *Neural Computing & Applications*, 1–7.

Shobha Rani & Pounambal. (2019). Deep learning based dynamic task offloading in mobile cloudlet Environments. *Evolutionary Intelligence,* 1-9.

Stavrinides, G. L., & Karatza, H. D. (2017). Scheduling real-time bag-of-tasks applications with approximate computations in SaaS clouds. Wiley Online Library.

Sui, X., Li Li, D. L., Wang, H., & Yang, H. (2019). Virtual machine scheduling strategy based on machine learning algorithms for load balancing. *EURASIP Journal on Wireless Communications and Networking*, *2019*(1), 1–16. doi:10.1186/s13638-019-1454-9

Tong, Deng, Hongjian, & Liu. (2019). QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment. *Neural Computing and Applications,* 1-18.

Waschneck, B., Reichstaller, A., Belzner, L., Altenmuller, T., Bauernhansl, T., Knapp, A., & Kyek, A. (2018). Optimization of Global Productin Scheduling with Deep Reinforcement Learning. *51st CIRP Conference on Manufacturing Systems, ScienceDirect*, 1264-1269.

Wu, Liu, & Zhao. (2018). An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem. *Memetic Computing,* 1-21.

Zang, Wang, Song, Lu, Li, Wang, & Zhao. (2019). Hybrid Deep Neural Network Scheduler for Job-Shop Problem Based on Convolution Two-Dimensional Transformation. *Computational Intelligence and Neuroscience,* 1-20.

Zhang, Ma, Xiao, Li, & Lin. (2019). *Two-level task scheduling with multi-objectives in geo-distributed and large-scale SaaS cloud.* Springer Nature.

Zhang, P. Y., & Zhou, M. C. (2017). Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy. *IEEE Transactions on Automation Science and Engineering*, 1–12.

1Zheng, X., Li, M., & Guo, J. (2020). Task scheduling using edge computing system in smart city. Wiley Online Library.

Zhou, Li, Zhu, Xie, Abawajy, & Chowdhury. (2018). An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. *Neural Computing and Applications,* 1-19.

Supriya Sawwashere is pursuing Ph.D. in Computer Science Engineering from Kalinga University, Raipur. Currently she is working in J. D. College of Engineering & Technology, Nagpur. She has teaching experience of 13.5 years. She has completed his M.Tech. in Computer Science Engineering from G.H. Raisoni College of Engineering, Nagpur (Maharashtra), India. She has completed his B.E. in Information Technology. She has published 8 papers in international journals and 3 papers in international conferences. Her areas of research interest are network security, theory of computation, computer graphics, and IoT.