# Resource Management in Cloud Computing Using Machine Learning: A Survey

Sepideh Goodarzy
*University of Colorado Boulder*
Sepideh.Goodarzy@colorado.edu

Maziyar Nazari
*University of Colorado Boulder*
Maziyar.Nazari@colorado.edu

Richard Han
*University of Colorado Boulder*
Richard.Han@colorado.edu

Eric Keller
*University of Colorado Boulder*
Eric.Keller@colorado.edu

Eric Rozner
*University of Colorado Boulder*
Eric.Rozner@colorado.edu

***Abstract*—Efficient resource management in cloud computing research is a crucial problem because resource over-provisioning increases costs for cloud providers and cloud customers; resource under-provisioning increases the application latency, and it may violate service level agreements, which eventually makes cloud providers lose their customers and income. As a result, researchers have been striving to develop optimal resource management in cloud computing environments in different ways, such as container placement, job scheduling and multi-resource scheduling. Machine learning techniques are extensively used in this area. In this paper, we present a comprehensive survey on the projects that leveraged machine learning techniques for resource management solutions in the cloud computing environment. At the end, we provide a comparison between these projects. Furthermore, we propose some future directions that will guide researchers to advance this field.**

***Index Terms*—resource management, cloud computing, machine learning**

## I. Introduction

Nowadays, industry and academia are moving their applications to the cloud. Cloud computing provides a platform to application developers so that they can run their applications in the cloud without worrying about server setups and configurations. On the other hand, cloud providers are constantly looking for ways to offer better services to the developers while they consider efficiency.

Resource management is the allocation of resources such as CPU, Memory, Storage and Network Bandwidth to the virtualization unit, such as virtual machines or containers, in the cloud. There is no finite outline for proper resource management. Good resource management can vary between cloud providers based on their primary aspirations. The main objectives usually are minimizing job completion time, makespan time, increasing resource efficiency, cost reduction, and energy optimization.

The objectives mentioned above are very significant. For example, Amazon report shows that it loses $1\%$ of its revenue for a 100ms increase in a response delay. Google discovered that an extra 0.5 seconds delay in search generation in Google will drop its traffic by $20\%$ [1]. This report shows the importance of Service Level Objective (SLO) preservation. According to [2], approximately $70\%$ of total data center operating costs are due to power, which shows the importance of minimizing energy consumption.

Researchers have been trying to propose solutions in this area by using different methods such as heuristics and algorithms. The problem with these methods is that they are not workload-specific, they are not capable of handling workload changes because they do not provide dynamicity, and they need prior knowledge of the workload to tune parameters. However, some of the researchers tried machine learning to solve this. Because it is workload-specific, it can handle dynamicity in workload behavior, and it does not need any workload specialization.

In this paper, we review some of the works in the literature that they proposed to manage and provision resources using machine learning, and we compare these solutions based on the methods they used, their objectives, their novelties, and their final results. We also investigate some of the possible future directions that researchers can study to improve state-of-the-art techniques.

There are a few surveys done in this area. For instance, in this recent survey [3] they provided a comprehensive study of resource management, but they are more focused on research papers in the context of performance management in the cloud and do not give a clear picture to the readers about the correlation between ML techniques and cloud computing-specific objectives. In another survey [4], they studied resource management with the focus on energy consumption optimization, which is not our main focus in this paper. However, unlike those, this paper's focus is more on providing a literature review categorized based on the ML techniques that are used, and comparing them. As a result, after reading this survey, the reader will have a general idea of why researchers chose different ML techniques for specific goals, what the drawbacks of their approaches are and how they addressed previous works' gaps. Hopefully, this will give a clear picture of the state-of-the-art research in using ML techniques in cloud computing resource management.

The paper is organized as follows. In section II, we study some reputable papers in the area. These papers are divided

based on their applied ML method to three categories: supervised, unsupervised and reinforcement learning, which are investigated in detail in sections II-A, II-B and II-C, respectively. In section III, we present a comparative overview of all the research reviewed in section II, and we recommend some possible future directions in this area. Section IV summarizes this paper.

## II. Survey

Three popular types of machine learning models used in resource management research are supervised, unsupervised, and reinforcement learning (RL). In supervised learning, the training dataset contains features and labels, and the model learns to predict the label considering the features. There is no label in the dataset in an unsupervised learning model, so the model discovers the relation between the training data and then find the correct labels based on the relation. In RL, an agent gathers information about the environment called state. Then, it does an action that changes the environment to the next state and returns a reward to the agent. This reward shows how much the agent's action optimizes the RL objective function. RL exploits state, action, and reward space to learn the best action sequence, which gains the most cumulative reward. We study the works that used supervised, unsupervised and reinforcement learning in Section II-A, II-B and II-C respectively.

### A. Supervised Learning

In [5], there is a performance model predictor in their system, which predicts a 5 min workload (req/sec) using the recent 15 minutes of workload. Smoothing splines [6] are used for the performance model predictor. Using the predicted workload, they estimate the number of required servers and the fraction of requests that violate the Service Level Agreement (SLA). In order to prevent oscillation in the number of required servers, a hysteresis parameter is employed. Whenever the system detects a change in performance model predictor performance, the system will switch to exploration mode in which it does online training and tries to adjust the model. In the exploration mode, at first, the system sets the required servers to the maximum available servers, then it gradually removes servers to reach the optimum number of required servers. Their solution does not cover maximizing SLAs besides minimizing the number of required servers objective; instead, they only try to minimize the number of required servers and not violate the SLA thresholds. As for benchmarking, they used the Cloudstone Web 2.0 benchmark [7], which has a workload generator called Faban [8], and they used real workload data from Ebates.com.

Unlike the previous work, [9]'s goal is to maximize SLA while optimizing energy consumption. They try to consolidate tasks to turn off unused machines. In order to handle turning on machines when the load increases, booting delay is considered. First, they predict the CPU Usage percentage of a task using linear regression to predict its power consumption. M5P is used to predict power consumption from CPU usage percentage prediction. Linear regression is also employed to predict the SLA using features like CPU usage, the time a job has spent so far, and available CPU. The paper assumes that all machines are identical. The authors used their predicted results in the Dynamic Backfilling to schedule jobs. Moving a job is based on interference with the required resource of other tasks running on the machine and trade-off between power consumption savings and performance degradation. In the evaluation section, a simulation is perfomed using OMNet++ [10]. To choose thresholds for turning off/on a machine, different min and max usage thresholds are tested. Because SLA guarantees are taken into account with a higher priority in scheduling jobs rather than minimizing energy consumption. their method is much less likely to violate SLAs. As for future work, they proposed to use reinforcement learning instead of the costly dynamic backfilling. It would be interesting to consider different machine types or other resources such as memory and network in their prototype. It would be more realistic if more metrics are considered in their simulation as well.

Instead of naively turning machines on and off, [11] has considered different on/off states in this work, and they chose the best on/off to offer lesser transition delay and power consumption. The feed-forward Neural Network (NN) is utilized to predict future workload based on workload history to decide on when it is suitable to turn off machines to save energy. For the evaluation, their system is simulated by using the CloudSim and GridSim toolkits [12], [13]. They generated workloads based on traces containing requests to NASA and ClarkNet [14]. The power consumption and requests' drop rate is measured in different configurations. Their best result is when they keep 20% extra servers on in addition to the predicted number of required servers. They did not compare their solution with any other power management scheme. It could be beneficial to propose a more general solution that will work with different workloads and data center architectures. It will be more realistic to test their method in a real-world environment and not keep 20% extra servers on.

The work [15] predicts the CPU utilization based on the recent CPU utilization using both Error Correction Neural Network and Linear Regression with/without sliding window technique [16]. They used the predicted results to turn on/off the unused VM to minimize the number of VMs. In order to find the best window size, different values were experimented with. The TPC-W benchmark [17] was employed to generate the workload. The authors have shown that their system can predict the future resource requirement surge sooner than the VM instance setup. This approach was not implemented and evaluated on the public cloud.

In [18] the primary focus is on cloud-based media processing. The main objective is to prevent QoS degradation and efficient resource management by minimizing the number of VMs. To minimize the number of VMs, they predict the CPU resource usage of a task based on similar previous tasks using Support Vector Regression (SVR). These tasks are placed on VMs based on the predicted results. The prediction confidence

metric using k-nearest neighbors (KNN) is employed to see if their prediction is reliable. If it is not reliable, they will run the task in a lightweight VM. To prevent QoS degradation, a survival function (Q function [19] ) is used to determine if combinations of tasks still obey QoS limits with a specified confidence level. It is good to add more dimensions (Memory, Network) to this problem as future work. Also, rearranging the existing tasks instead of only placing them can make a significant improvement. Their work also lacks consideration of the variety of VMs and jobs like CPU bound or I/O bound tasks.

### *B. Unsupervised Learning*

In DejaVu [20], each service has a proxy responsible for forwarding the users' requests to the profiler. The profiler computes each workload's signatures, consisting of low-level metrics to provide non-intrusive and low overhead monitoring. These signatures are used for clustering the workloads using K-means. The required resources for each cluster are computed via linear search. After the required resource is computed for a workload, the tuner maps that workload to a virtualized resource. When DejaVu detects changes in the workload, it will classify the workload using the C4.5 decision tree based on its VM Id and its signature. When the classification certainty level is low, DejaVu sets the workload to its full capacity configuration and clusters and tunes again. They compute performance interference by comparing the workload performance in the profiler with the production, and they cover this issue by providing more resources to the corresponding service. For the evaluation, different services were tested such as SPECWeb2009 [21], Cassandra [22] and RUBiS [23] by replaying MSN messenger and HotMail traces [24].

### *C. Reinforcement Learning*

In paper [25], reinforcement learning (SRASA(0)) is used for offline training combined with different queuing models [26]–[29] for online training to prevent the poor performance of RL in the beginning [25]. Their RL model employs a Neural Network (NN) inside the RL in place of a lookup table to avoid the need to explore all the state, action space. The main objective is to maximize net expected revenue. Their method is tested through the TRADE3 application. They tested both open-loop traffic and closed-loop traffic models. In order to simulate a stochastic bursty time-varying demand, time series traffic is modified [30]. Also, a batch CPU intensive workload is tested as a sample of a non-web-based workload. Their model handles dynamicity such as transient and switching delays by including the previous decision in the input features. Future work could include expanding their action space from only server allocation to resource management ( CPU, Memory, Network bandwidth). They can also expand their state space from the mean page request arrival rate only, by adding other measurements.

In [31], the authors proposed a multi-resource cluster scheduler that schedules jobs online. The standard policy gradient reinforcement learning [32] is used combined with DeepNN [33] (rmsprop [34]). The assumption is made that jobs are nonpreemptable. As stated in their paper, two resource types are considered (though more types are supported) and the state space contains the current resource allocation in the cluster and the resource profiles of the first M jobs waiting to be scheduled as well as the number of other jobs stored in the backlog. DeepRM can schedule jobs based on an objective such as minimizing average job completion time or job slowdown, which can be chosen dynamically by defining the correct reinforcement rewards. As for the evaluation, they compared their solution with standard heuristics such as the shortest job first or a packing scheme inspired by Tetris [35]. Future work could add multiple machines [36] into their problem space instead of one large machine, which brings out the fragmentation problem. Data locality [37], [38] and inter-task dependencies [35] could also be considered. They assumed that a job's resource requirement is known beforehand, but that may not be the case for non-recurring jobs [39]. RL can solve this partial visibility by casting the decision problem as a POMDP [40]. Their learning phase is based on a finite time horizon, which can be solved using a value network [41], that estimates the average return value.

Decima [42] is an automatic, highly efficient, workload-specific, and general-purpose scheduler for data processing jobs to minimize job completion time. They used RL and graph neural networks [44]–[47] in their model besides directed acyclic graphs (DAG) [48]–[51] to represent job dependency graphs. Their method converts DAGs of different sizes and shapes to input vectors to the policy network. Each job stage is scheduled with an efficient parallelism level. To deal with continuous streaming of job arrivals, they terminate the training episodes early in the beginning and gradually grow their length, making policy to learn how to handle short and straightforward job sequences and then learn more lengthy challenging sequences [52]. For handling stochastic job arrivals, the variance reduction technique [53], [54] is employed. In the evaluation, Decima is integrated with Spark [55], and workload traces are used from Alibaba and tested out in Spark clusters [56], [57]. Other techniques include handling multi-resource scheduling, batch mode, and resource fragmentation. In terms of future work, they can use robust adversarial RL [58] for drastic workload changes and meta-learning [59]–[61] for online learning when a workload change happens. Multi-agent RL [60]–[62] could be utilized to handle preemptable jobs such that one agent is responsible for scheduling the next stage, and the other one decides on executors to be preempted.

In [43], the paper used a decentralized deep-Q-network in a multi-agent RL setting to schedule multi-workflow in IaaS clouds with heterogeneous VMs with different configurations and pricing models. Their main goals are minimizing make-span time and user's cost optimization. Their work's novelty is that their solution can reach correlated equilibrium policy in a dynamic real-time environment to optimize more than one objective using multi-agent reinforcement learning. They also considered workflow DAGs, and they learned the correct parallelism level for each task. Well-known scientific workflows

TABLE I
AN OVERVIEW OF ML PROPOSALS TO MANAGE RESOURCES IN CLOUD COMPUTING

| Year | Ref. | ML Tech | Dataset | Features | Output | Goal |
|---|---|---|---|---|---|---|
| 2009 | Bodik [5] | Supervised: Smooth Splines and LOESS | Cloudstone web 2.0 bencmark, Faban workload generator, Ebates.com traces | Recent workload, #estimated required server | Mean performance and performance variance | Minimiz #servers |
| 2010 | Berral [9] | Supervised: Linear regression, M5P | Simulation: OMNet++ | Recent CPU usage percent, available CPU, the time the job has spent so far | CPU usage percent, power consumption, SLA | Maximize SLA while optimizing energy consumption |
| 2010 | Duy [11] | Supervised: Feedforward NN | Simulation: CloudSim, GridSimNASA and ClarkWeb traces | Recent workload | #Required Servers | Minimiz #servers |
| 2012 | Islam [15] | Supervised: Error corection NN and LR with/without window size technique | TPC-W benchmark | Cpu utilaztion history | Cpu utilaztion | Minimiz #VMs |
| 2013 | Sembiring [18] | Supervised: SVR, KNN, | An script of media tasks | Media Task features | Cpu utilaztion | Minimiz #VMs |
| 2012 | Vasic [20] | Unsupervised: Clustering | SPECWeb2009, Cassandra and RUBiS servisedMSN messenge, HotMail traces | Workoad low level Signature | Workload cluster | Meeting SLO when a workload change happens |
| 2006 | Tesauro [25] | RL:SARSA(0) combined with NN | TRADE3 | State: Workload and performance | Action: server allocation | Maximize net expected revenue (which is based on performance-based SLA) |
| 2016 | Mao [31] | RL with DeepNN (rmsprop) | Their Simulation | State: The current resource allocation, the resource profiles of the first M jobs waiting to be scheduled, #jobs stored in the backlog | Action: scheduling jobs | Minimizing average job completion time |
| 2019 | Mao [42] | RL with Graph Neural Network | Alibaba's traces on Spark cluster | State: Jobs' DAGs | Action: Scheduling jobs with the efficient level of parallelism | Minimizing job completion time |
| 2019 | Wang [43] | Multi-agent RL: Decentralized deep-Q-network | Well-known scientific workflows | State: Jobs' DAGs | Action: Scheduling jobs with the efficient level of parallelism | Minimizing make-span and user's cost |

are employed with different tasks on Amazon EC2 instances for evaluation. Future work can consider more than two QoS metrics, such as reliability, and load balancing. On-the-fly scheduling could also be explored.

## III. DISCUSSION

Table I presents an overview of all the works that are mainly discussed in detail in this paper. The majority of papers used supervised learning. Most of the works that leveraged supervised learning have employed recent workloads in order to predict the current/future workload. Their main objectives are minimizing the number of servers or VMs to save energy and decrease costs. The VM/server in which a job is going to be scheduled should have enough resources. Adding the new job to it should not interfere with other jobs residing on that VM/server. The ultimate VM/server among all the VMs/servers that satisfy the above conditions is selected in a greedy manner, which does not always guarantee a return to the most optimum choice.

After supervised learning, reinforcement learning is the next most prevalent method explored [63]. The state-space in most RL works consists of jobs' DAGs, and the action space is scheduling jobs and specifying the level of job parallelism. Their main objective is mostly to minimize job completion time. The problem with these solutions is that their RL methods are not online, so whenever a change happens in the workload, there will be a delay in re-training the model and responding to that change. So, they should use another method besides RL for those situations.

Unsupervised learning is least suitable for resource management because it divides workloads into clusters, and many workloads end up in the same cluster. As a result, the system assigns the same resources for all the members of a cluster. The only exception where unsupervised learning happens to be beneficial is when the workload changes.

A promising direction for future work is for researchers to study online learning with RL. They can use meta-learning. It is going to be interesting to consider more than two conflicting objectives by having more than two agents, and researchers should also explore preemptible jobs by using a specified agent for handling job preemption decisions. It would be better if RL models did not rely on the job resource usage profile because they are not always accurate; instead, they should estimate each job resource profile using supervised learning.

## IV. CONCLUSION

In summary, in this paper, we provided a comprehensive survey on background works that have taken advantage of the machine learning techniques to solve real-world problems in the cloud computing area, and they have applied ML algorithms to optimize different objectives related to the cloud

computing environment. We also discussed the potential future directions of this research area, and we are hoping that using this literature review helps researchers make more progress in this field.

## V. Acknowledgment

This research was supported in part by VMware and the NSF as part of SDI-CSCS award number 1700527, and by the NSF as part of CAREER award number 1652698.

## References

[1] Y. Einav, *Amazon Found Every 100ms of Latency Cost them 1% in Sales*, 2019 (accessed September 25th, 2020). [Online]. Available: https://www.gigaspaces.com/blog/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/

[2] M. Rareshide, *Power in the Data Center and its Cost Across the U.S.*, 2017 (accessed September 25th, 2020). [Online]. Available: https://info.siteselectiongroup.com/blog/power-in-the-data-center-and-its-costs-across-the-united-states

[3] S. K. Moghaddam, R. Buyya, and K. Ramamohanarao, "Performance-aware management of cloud resources: A taxonomy and future directions," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–37, 2019.

[4] K. Braiki and H. Youssef, "Resource management in cloud data centers: a survey," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2019, pp. 1007–1012.

[5] P. Bodík, R. Griffith, C. A. Sutton, A. Fox, M. I. Jordan, and D. A. Patterson, "Statistical machine learning makes automatic control practical for internet datacenters." *HotCloud*, vol. 9, pp. 12–12, 2009.

[6] J. L. Hellerstein, V. Morrison, and E. Eilebrecht, "Optimizing concurrency levels in the. net threadpool: A case study of controller design and implementation," *Feedback Control Implementation and Design in Computing Systems and Networks*, 2008.

[7] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, A. Fox, and D. Patterson, "Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0," in *Proc. of CCA*, vol. 8, 2008, p. 228.

[8] "Next generation benchmark develop- ment/runtime infrastructure," 2008. [Online]. Available: http://faban.sunsource.net

[9] J. L. Berral, Í. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, and J. Torres, "Towards energy-aware scheduling in data centers using machine learning," in *Proceedings of the 1st International Conference on energy-Efficient Computing and Networking*, 2010, pp. 215–224.

[10] "Omnet++," 2009. [Online]. Available: http://www.omnet.org

[11] T. V. T. Duy, Y. Sato, and Y. Inoguchi, "Performance evaluation of a green scheduling algorithm for energy savings in cloud computing," in *2010 IEEE international symposium on parallel & distributed processing, workshops and Phd forum (IPDPSW)*. IEEE, 2010, pp. 1–8.

[12] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *2009 international conference on high performance computing & simulation*. IEEE, 2009, pp. 1–11.

[13] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation: practice and experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[14] "Tracesintheinternettrafficarchive." [Online]. Available: http://ita.ee.lbl.gov/html/traces.html

[15] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[16] T. G. Dietterich, "Machine learning for sequential data: A review," in *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*. Springer, 2002, pp. 15–30.

[17] "Tpc, tpc-w benchmark, transaction processing performance council (tpc)," San Francisco, CA 94129-0920, USA, 2003.

[18] K. Sembiring and A. Beyer, "Dynamic resource allocation for cloud-based media processing," in *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2013, pp. 49–54.

[19] M. K. Simon, *Probability distributions involving Gaussian random variables: A handbook for engineers and scientists*. Springer Science & Business Media, 2007.

[20] N. Vasić, D. Novaković, S. Miučin, D. Kostić, and R. Bianchini, "Dejavu: accelerating resource allocation in virtualized environments," in *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems*, 2012, pp. 423–436.

[21] "Specweb2009." [Online]. Available: http://www.spec.org/web2009/

[22] "Apache foundation. the apache cassandra project." [Online]. Available: http:// cassandra.apache.org/

[23] E. Cecchet, J. Marguerite, and W. Zwaenepoel, "Performance and scalability of ejb applications," *ACM Sigplan Notices*, vol. 37, no. 11, pp. 246–261, 2002.

[24] E. Thereska, A. Donnelly, and D. Narayanan, "Sierra: practical power-proportionality for data center storage," in *Proceedings of the sixth conference on Computer systems*, 2011, pp. 169–182.

[25] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, "A hybrid reinforcement learning approach to autonomic resource allocation," in *2006 IEEE International Conference on Autonomic Computing*. IEEE, 2006, pp. 65–73.

[26] P. Pradhan, R. Tewari, S. Sahu, A. Chandra, and P. Shenoy, "An observation-based approach towards self-managing web servers," in *IEEE 2002 Tenth IEEE International Workshop on Quality of Service (Cat. No. 02EX564)*. IEEE, 2002, pp. 13–22.

[27] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements," in *International Workshop on Quality of Service*. Springer, 2003, pp. 381–398.

[28] M. N. Bennani and D. A. Menasce, "Assessing the robustness of self-managing computer systems under highly variable workloads," in *International Conference on Autonomic Computing, 2004. Proceedings.* IEEE, 2004, pp. 62–69.

[29] ——, "Resource allocation for autonomic data centers using analytic performance models," in *Second international conference on autonomic computing (ICAC'05)*. IEEE, 2005, pp. 229–240.

[30] M. S. Squillante, D. D. Yao, and L. Zhang, "Internet traffic: periodicity, tail behavior, and performance implications," in *System performance evaluation: methodologies and applications*, 2000, pp. 23–37.

[31] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 2016, pp. 50–56.

[32] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[33] H. D. Beale, H. B. Demuth, and M. Hagan, "Neural network design," *Pws, Boston*, 1996.

[34] G. Hinton, N. Srivastava, and K. Swersky, "Overview of mini-batch gradient descent," *Neural Networks for Machine Learning*, vol. 575, 2012.

[35] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource packing for cluster schedulers," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 455–466, 2014.

[36] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*, 2013, pp. 1–16.

[37] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: fair scheduling for distributed computing clusters," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, 2009, pp. 261–276.

[38] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in *Proceedings of the 5th European conference on Computer systems*, 2010, pp. 265–278.

[39] S. Agarwal, S. Kandula, N. Bruno, M.-C. Wu, I. Stoica, and J. Zhou, "Reoptimizing data parallel computing," in *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 2012, pp. 281–294.

[40] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 362–370.

[41] R. S. Sutton, A. G. Barto *et al.*, "Introduction to reinforcement learning. vol. 135," 1998.

[42] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 270–288.

[43] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie, "Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 39 974–39 982, 2019.

[44] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[45] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 6348–6358.

[46] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[47] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[48] "Apachetez2013.apachetezproject," 2019. [Online]. Available: https://tez.apache.org/

[49] C. Chambers, A. Raniwala, F. Perry, S. Adams, R. R. Henry, R. Bradshaw, and N. Weizenbaum, "Flumejava: easy, efficient data-parallel pipelines," *ACM Sigplan Notices*, vol. 45, no. 6, pp. 363–375, 2010.

[50] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, 2007, pp. 59–72.

[51] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 2012, pp. 15–28.

[52] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[53] H. Mao, S. Chen, D. Dimmery, S. Singh, D. Blaisdell, Y. Tian, M. Alizadeh, and E. Bakshy, "Real-world video adaptation with reinforcement learning," 2019.

[54] H. Mao, S. B. Venkatakrishnan, M. Schwarzkopf, and M. Alizadeh, "Variance reduction for reinforcement learning in input-driven environments," *arXiv preprint arXiv:1807.02264*, 2018.

[55] "Apache spark: Dynamic resource allocation," 2018, spark v2.2.1 Documentation. [Online]. Available: http://spark.apache.org/docs/2.2.1/job-scheduling.html#dynamic- resource- allocation

[56] "Cluster data collected from production clusters in alibaba for cluster management research," 2017. [Online]. Available: https://github.com/alibaba/clusterdata

[57] C. Lu, K. Ye, G. Xu, C.-Z. Xu, and T. Bai, "Imbalance in the cloud: An analysis on alibaba cluster trace," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 2884–2892.

[58] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2817–2826.

[59] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-based reinforcement learning via meta-policy optimization," *arXiv preprint arXiv:1809.05214*, 2018.

[60] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[61] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1126–1135.

[62] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

[63] Y. Zhang, J. Yao, and H. Guan, "Intelligent cloud resource management with deep reinforcement learning," *IEEE Cloud Computing*, vol. 4, no. 6, pp. 60–69, 2017.