Simulation Runtime Prediction Approach based on Stacking Ensemble Learning

Yuhao Xiao, Yiping Yao, Feng Zhu* and Kai Chen
College of Systems Engineering, National University of Defense Technology, Changsha 410073, China

Keywords: Complex System Simulation, Cloud Computing, Runtime Prediction, Ensembling.

Abstract: Many technologies in complex system simulation (CSM), such as resource scheduling and load balancing, largely rely on historical data with different characteristics to predict the future. The accuracy of runtime prediction has a significant impact on scheduling performance. However, with cloud computing becoming the main infrastructure for deploying CSM applications, the current prediction methods are difficult to adapt to the dynamic changes of cloud computing resources. Insufficient computing resource allocation will be difficult to support the efficient operation of simulation. In addition, excessive computing resource allocation will lead to higher computing and data communication costs. Therefore, a simulation runtime prediction approach based on stacking ensemble learning has been proposed, which uses the characteristic variables of simulation applications (such as the number of simulation entities, the number of simulation events, the simulation time, etc.) and the performance monitoring data of computing resources (such as CPU utilization, memory utilization, etc.) as the characteristic inputs. The machine learning algorithms such as XBG, SVG, MLP are integrated by stacking model, and the performance of the integrated learning algorithm is comprehensively evaluated by mean absolute error (MAE), accuracy (ACC), root mean square error (RMSE) and coefficient of determination (R2). Experimental results show that the proposed algorithm improve the prediction accuracy by 3% - 24% when compared with existing machine learning prediction methods.

1 INTRODUCTION

Complex system simulation (CSM) is widely used in system evaluation and analysis in the fields of social and biological networks, military training, and war games (Fujimoto et al. , 2017; Haken, 2006). In CSM, the simulation models are partitioned onto logical processes (LPs, also called simulation entities) and exploits the parallelism present in the LPs to improve the performance. With the increase in the scale of CSM applications, interaction between entities has become very complex, which leads to irregular workloads and communication loads, and reduces the operating efficiency of the underlying infrastructure (Fujimoto, 2016). In addition, in order to cope with the huge workload changes and to reduce simulation runtimes, a powerful computing infrastructure is required for perform efficiently. Unlimted computing resources for most organizations are difficult considering budget constraints. Therefore, a flexible infrastructure deployment mechanism is needed, which can allocate resources according to the computing requirements of applications.

Cloud computing provides a good target environment for the above challenges, which can realize the collaborative management, on-demand sharing, and flexible scheduling of resources such as computing/network/software/models, and meet the requirements of CSM for general computing power and efficient operation (Fujimoto, Malik, & Park, 2010). In order to utilize this environment, we monitor and collect application performance in real time during the execution of simulation applications, so as to perform real-time resource scheduling in the cloud environment and reduce the overall deployment cost (D'Angelo & Marzolla, 2014).The resource scheduling process largely relies on the estimation of task runtime to make decisions (D'Angelo & Marzolla, 2014). However, traditional methods are difficult to predict the runtime of applications in the cloud environment because of the continuous requirements changes for computing resources caused by the different life cycles of simulation entities. If too few resources are allocated to the application, it will be difficult to support the efficient operation of the application; if too many resources are

Xiao, Y., Yao, Y., Zhu, F. and Chen, K.
Simulation Runtime Prediction Approach based on Stacking Ensemble Learning.
DOI: 10.5220/0010517600420049
In *Proceedings of the 11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2021)*, pages 42-49
ISBN: 978-989-758-528-9

allocated to the application, it will not only increase the communication overhead between different node entities, but also cause unnecessary waste of resources (Liu et al., 2012). Therefore, how to accurately predict the execution time of simulation application in a cloud environment and achieve efficient resource allocation of complex system simulation application is a challenging task.

In response to the above problems, this paper models the prediction problem as a regression function that depends on the static feature input before the simulation application runs and the dynamic feature input at run time, and proposes an integrated learning method based on stacking to predict the execution time of the simulation application in a specific cloud computing environment, so as to achieve efficient resource management for the cloud simulation application.

The main contributions of this article are as follows:

1. A set of parameters is designed to represent the characteristics of simulation application in cloud. The prediction process uses the following parameters as inputs: (1) the characteristic parameters that need to be set before simulation execution (such as the number of simulation entities, the number of events, etc.) and (2) the execution parameters collected by the monitor (such as CPU utilization, memory utilization, etc.).

2. A simulation runtime prediction algorithm based on stacking ensemble learning (SRPAS) is proposed. Compared with linear regression, multi-layer perceptron, regression tree, random forest and other machine learning methods, our algorithm can improve the accuracy and reduce the error rate.

The rest of this paper is arranged as follows: Section 2 introduces related work. In Section 3, the proposed prediction algorithm is introduced. Section 4 introduces the experiments and results of this work. Section 5 introduces conclusions and future work.

2 RELATED WORK

Cloud-based simulation is regarded as a special Software-as-a-Service (SaaS) that combines the advantages of Web services and cloud computing technology to manage various M&S resources and build different simulation environments(Bocciarelli et al., 2017). However, the uncertainty of resource sharing in cloud environment may lead to the degradation of simulation application performance (Kurt, Silas, & Jan, 2012). To solve this problem, it is recognized that a better solution is to use machine learning prediction methods to improve the adaptability of simulation applications in the cloud environment. For example, linear regression LR (Lee & Schopf, 2004; Seneviratne & Levy, 2011), nearest neighbor KNN (Hui, Groep, & Wolters, 2005), regression tree (Miu & Missier, 2012), support vector machine (Matsunaga & Fortes, 2010), extreme gradient boosting (Chen, He, & Benesty, 2016) and so on. Yang (Yang et al., 2014) proposed a linear regression model to predict the workload in the cloud, and proposed a greedy heuristic algorithm to achieve the scalability of cloud services. Cetinski (Cetinski & Juric, 2015) proposed an advanced model for efficient workload prediction in the cloud environment. By analyzing the characteristics of the workflow in the cloud, the random forest algorithm is applied to solve the workload prediction problem. Rahmanian (Rahmanian et al., 2018) proposed a prediction algorithm based on learning automata to predict the use of cloud resources, and experiments on the CoMon project data set proved the effectiveness of the algorithm. In order to cope with the ever-changing cloud resource demand, Kaur(Kaur, Bala, & Chana, 2019) proposed an intelligent regression integrated prediction method, which combines feature selection and resource prediction technology to obtain high accuracy. Kim (Kim, Wang, Qi, & Humphrey, 2020) designed an online mechanism to predict workflow load in a cloud environment. This mechanism accurately estimates the relative accuracy of each predictor in the next time interval by using multi-class regression, so as to accurately predict the actual workload.

Although some models have been used to predict the runtime of applications, most of these are for highly parallelizable tasks (such as parallel tasks, video, etc.) in the cloud. Due to the simulation application will perform frequent synchronization during execution and send/receive a large number of messages, it is difficult to decompose the simulation application into independent subtasks. On the one hand, these prediction methods do not consider the characteristics of simulation applications in the cloud environment; On the other hand, the method based on a single predictor is not enough to solve the dynamic and burst of cloud workload, and may show poor performance for unknown workload patterns. Therefore, based on the versatility of the integrated model in this field, this paper aims to design an integrated algorithm to predict the runtime of simulation applications in the cloud environment and improve the performance of the simulation applications in the cloud environment.

3 THE PROPOSED PREDICTION APPROACH

It is well known that the combination of multiple models' predictions performs better than a single model, and usually significantly reduces the variance of the final prediction results (Dietterich, 2000). Ensemble learning involves merging multiple prediction results from different learning algorithms to create stronger predictor and achieve more accurate forecast results. Figure 1 shows the implementation process of the proposed simulation runtime prediction algorithm based on stacking ensemble learning (SRPAS).

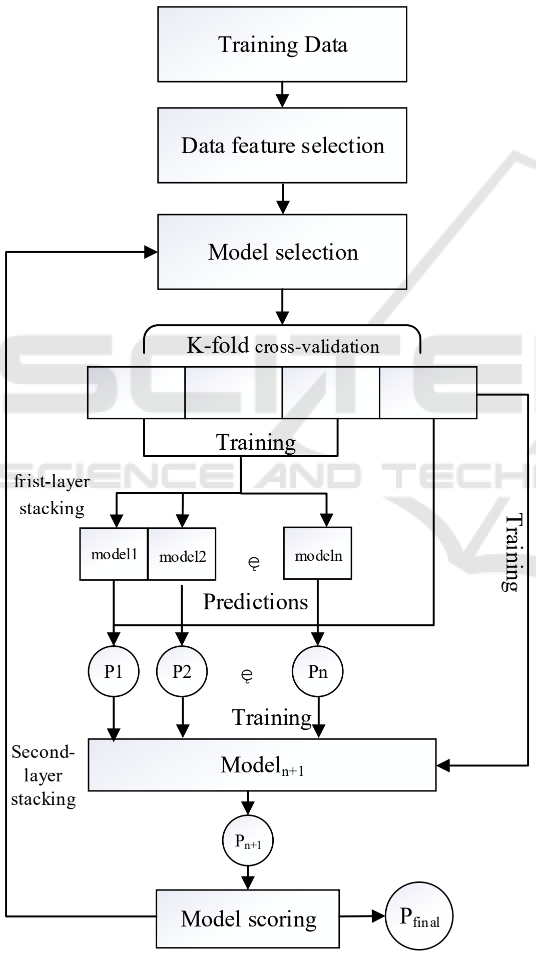

Figure 1: The schematic diagram of the runtime prediction algorithm.

3.1 Dataset Generation

To monitor and collect the real-time running information of the simulation application accurately, the resource monitoring program is deployed on the cloud computing nodes, and two sets of parameters are considered to evaluate the performance of the application:(1) pre-run parameters and (2) run-time parameters. The pre-run parameters are determined before the execution of the simulation application, including the number of computing nodes, the number of simulation entities and events, the look ahead value and the simulation times. Run-time parameters reflect the performance of simulation applications under different cloud resources. The parameters are collected during the execution of simulation applications, including CPU utilization, memory utilization, network throughput, network latency, and file system utilization. The monitor collects data every 5 seconds and stores it into the database as feature data after preprocessing. Finally, the research problem turns into how to combine these feature data with machine learning model to accurately predict the runtime of simulation. The specific parameters are shown in Table 1.

Table 1: Parameter description.

parameters	Feature	Description
Pre-run parameters	CPU core	CPU cores allocated for a simulation application.
	Entity	Number of simulation objects
	events	Number of events generated by per simulation object
	LA	Lookahead
	Tend	Simulation runtime
runtime parameters	CPU usage	Percentage of CPU (including average, maximum and minimum)
	Memory	Bytes of memory (including average, maximum and minimum)
	I/O	File system usage (including average, maximum and minimum)
	Network-r	Bytes of data received by the network(including average, maximum and minimum)
	Network-s	Bytes of data sent by the network(including average, maximum and minimum)
	Network delay	Communication delay

3.2 Selection of Prediction Model

Single machine learning model has limitations in meeting the requirements of complex tasks. By combining multiple basic models, the integrated model can achieve better performance than single model (Cruz, Sabourin, Cavalcanti, & Ren, 2015). However, not all basic models have a positive impact on the performance of the integration model. Therefore, we use 10 kinds of machine learning prediction models automatically adjusted by gridsearch, and find the model combination with the highest score (R2) based on the model selection process. The specific model and parameters are shown in Table 2.

Table 2: Machine Learning Models.

Model	Method Used	Tuning Parameters
ETR	ExtraTree	n_estimators=100, max_depth=None
KNG	K-Nearest Neighbor	n_neighbors=7
LR	linear regression	alpha=0.5
RF	Random Forest	n_estimators=100, max_depth=None
XBG	Extreme Gradient Boosting	learning_rate=0.1, max_depth=4 n_estimators=200
DT	DecisionTree	max_depth=5
MLP	Multi-layer Perceptron	hidden_layer_sizes=(100, 50), activation='relu', solver='adam', max_iter=1000
SVR	Support Vector Regression	kernel='poly', degree=2, C=1, epsilon=0.01
BRR	Bridge Regression	T = 1000, lambda2 = 1
NN	Neural Network	maxit=100, axNWts=10000

3.3 Proposed Algorithm

The integrated model can overcome the shortcoming of single basic prediction, and use the interaction between basic models to improve the prediction ability (Van der Laan, Polley, & Hubbard, 2007). The SRPAS proposed in this paper can select the optimal subset of the base model according to the data characteristics, and further improve the prediction accuracy of the integrated model. The specific steps are shown in algorithm 1.

In this algorithm, ModelList contains the list of all basic models, ModelSet represents the list of models being used for ensemble learning, ModelSet $\in$ ModelList. BestMSet and BestMScore respectively represent the best model combination and the highest model score based on the current ModelSet.

Firstly, Random strategy is used to select a model combination from the ModelList, and the stacking integration training is started.

Lines 1 to 9 show that all models in the first layer are trained by k-fold cross-validation, and the predicted value P^j of each model M is output.

Lines 10 to 12 represent the second layer of stacking integration process. In this process, the predicted value of the first layer model is used as the feature, and the final integrated model is trained after fusion with the original feature.

Lines 13 to 17 represent the model selection process. The process evaluates the integrated model based on the R^2 value. If the score of the current model combination is greater than the best model score, the best model combination and the best model score will be replaced. Finally, the algorithm returns the best model combination and the best prediction result as the output.

3.4 Evaluation Metrics

Mean Absolute Error (MAE), Accuracy (ACC), Root Mean Squared Error (RMSE) and Coefficient of Determination (R2) are calculated to evaluate the performance of resource prediction models, which are defined as follows:

$$MAE = \frac{\sum_{j=1}^{n} |y_r^j - y_P^j|}{n} \tag{1}$$

$$ACC = 1 - \frac{100\%}{n} \sum_{j=1}^{n} \frac{|y_r^j - y_P^j|}{y_P^j} \tag{2}$$

$$RMSE = \sqrt{\frac{\sum_{j=1}^{n} (y_r^j - y_P^j)^2}{n}} \tag{3}$$

$$R^2 = 1 - \frac{\sum_{j=1}^{n} (y_r^j - y_P^j)^2}{\sum_{j=1}^{n} (\bar{y_i} - y_r^j)^2} \tag{4}$$

Where y_r^j and y_p^j represents the real and predicted values of the j-th sample respectively, $\bar{y_r}$ is the average of the true values.

Algorithm 1: Proposed algorithm.

```
Input : Data(X,Y), ModelList M={M1, M2, ..., Mn}
Output : BestMSet, RuntimePred, BestMScore
1   For each modelset ∈ modellist
2     start {frist-layer-stacking}
3     Randomly split Data(X,Y) into k chunks $\{X^j, Y^j\}_{j=1}^{k}$
4       For j=1 to k do {k-fold bagging}
5         For each model m in M do
6           Training m-model on $\{(X^{-j}, Y^{-j}\}$
7           Make predictions $P^j$ on $X^j$
8         End for
9       End for
10    Choose model $M_i$ in M start {two-layer-stacking}
11    Train m model on $\{(X^j, P^j), Y^j\}$ and Make predictions $P_{final}$
12    End {stacking}
13    Caculate StackingMScore= $R^2$
14    If StackingMScore > BestScore then
15      bestscore← StackingMScore
16      BestMSet←ModelSet
17      $P_{final}$← $P_{final}$
18      End if
19  End for
20  return $\{M_{i1}, M_{i2}, \ldots M_{ir}\}$, $P_{final}$
```

4 EXPERIMENTAL STUDY

4.1 Simulation Application Settings

Phold is usually used as a representative benchmark test program for PDES performance evaluation (Yoginath & Perumalla, 2015). Therefore, the performance of the proposed algorithm is tested by the simulation application of Phold, and different simulation applications are generated by setting the parameters in Table3 to execute in the cloud environment.

4.2 Experimental Setup

Using real cloud environment monitoring data to track and predict the runtime of simulation applications will be more convincing. Therefore, this paper deploys the simulation application in a real cloud environment built by the open container engine docker, and uses container virtualization technology to build eight docker container nodes on two computing nodes. Each docker container node is configured with a 3.40 GHz Intel Core i5-7500 CPU core and 4GB of memory. In this paper, the simulation application is executed on YH-SUPE, and resources are reallocated for the simulation application according to the predicted results. The proposed prediction model is evaluated on two desktop computers with 3.40 GHz Intel (R) core (TM) i5-7500 CPU and 16GB RAM based on Python 3.5 programming environment.

4.3 Experimental Results and Analysis

4.3.1 Feature Selection

Feature selection, as a data preprocessing strategy, has been proven effective in data for various data mining and machine learning problems. The purpose of feature selection is to build a simpler and easier to understand model and improve the performance of the model (Tang, Alelyani, & Liu, 2014). Feature extraction generally applies mathematical methods to map data from high-dimensional feature space to low-dimensional feature space, which transforms the original features that cannot be recognized by machine learning algorithm into new features that can be recognized. However, we have observed that the data features in the simulation application log data set are all expressed in numerical form and do not need to be converted. Therefore, we can apply feature selection methods to remove irrelevant features, select strongly related features and weakly related but not redundant features to minimize the occurrence of errors and build a more accurate prediction model. in this paper, we use embedded feature selection method to rank the importance of features(Li et al., 2017). As shown in Figure 2, these features are input into the prediction model.

Table 3: Parameter configuration.

Parameters	value
Number of simulation objects	[10,200]
Number of events generated by per simulation object	[10,100]
Simulation run time	1000
Time synchronization strategy	Conservative
Lookahead	[0.1,1]

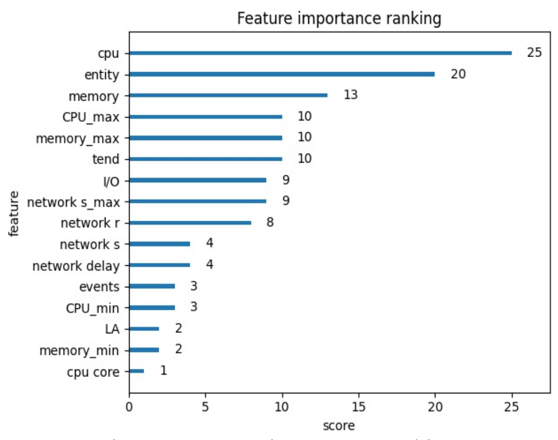

Figure 2: Feature importance ranking.

4.3.2 Model Performance Evaluation

In this part, the performance of SRPAS is evaluated. To evaluate the effectiveness of the algorithm in detail, we have carried out a large number of experiments. First, the performance of each machine learning model is measured, and then the integrated model is evaluated. The experimental results are shown in Table 4. SRPAS is the integration algorithm with model selection proposed in this paper, and eight base models are the best combination of models obtained by model scoring. SRPA is an integration algorithm without model selection process. In this algorithm, all prediction models trained in Table 2 will participate in the integration.

Table 4: Model performance evaluation.

Model	R^2	RMSE	MAE	ACC (%)
KNN	0.817	97.79	62.13	75.31
SVR	0.942	53.28	34.83	81.59
MLP	0.948	53.03	38.23	83.35
LR	0.946	54.61	37.50	83.37
DT	0.957	48.67	36.43	84.45
ETR	0.962	45.71	32.03	86.37
RF	0.963	42.17	28.17	89.81
XBG	0.966	40.01	28.30	90.21
SRPA	0.977	36.56	23.69	89.54
SRPAS	0.972	37.7	22.04	93.72

It can be seen from Figure 3, each model has different performance in any evaluation index. More precisely, a single prediction model may be better than other prediction models in terms of error rate, but it may have worse accuracy. For example, in Figure 3 (c), the accuracy of LR model is 83%, which is higher than that of SVR model, but the RMSE and Mae values are 54.61 and 37.50, which are also higher than the error rate of SVR model. In addition, this paper tests the integrated model without model selection process, and finds that SRPAS model has higher prediction accuracy (93.72%) when the error rate (37.7 / 22.04) is equivalent to SRPA model.

We set two groups of simulation events parameters for phold (50 and 100 respectively), and execute them in parallel with different number of CPU cores. As shown in Figure 4, the prediction results of proposed SRPAS is close to the real value, and the maximum error is ± 30 seconds. In addition, when the number of phold simulation events is set to 50 in Figure 4(a), the application has the shortest runtime in the case of allocating 7 cores. However, when the number of phold simulation events increases to 100 in Figure 4(b), the resource required for the shortest runtime are reduced to 6 cores.

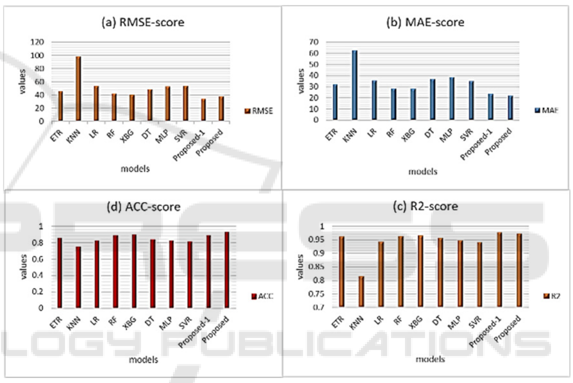

Figure 3: Evaluation Metrics.

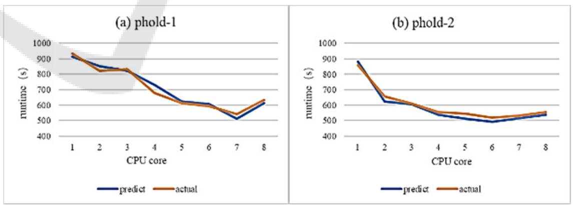

Figure 4: Comparison of actual simulation runtime and predicted results.

In general, the results show that the proposed algorithm can effectively predict the runtime of simulation applications and select the optimal computing resources for applications. Compared with the existing machine learning algorithms, SRPAS can improve the prediction accuracy by 3% - 24% while maintaining the minimum error rate. In the runtime prediction, SRPAS can select the basic model with better performance, effectively reduce the impact of poor performance model, and finally improve the prediction performance. Therefore, the simulation runtime prediction approach proposed in this paper is superior to the existing single machine learning regression model.

5 CONCLUSIONS

This paper discusses the runtime prediction programs for CSM in the cloud environment, we propose a simulation runtime prediction method based on ensemble learning to support the efficient deployment for CSM in the cloud. Firstly, the simulation application is deployed in the cloud environment to generate the data set, and the feature selection technology is utilized to obtain the relevant feature set. Secondly, a prediction algorithm based on stacking ensemble learning is proposed, which improves the prediction accuracy of ensemble model by selecting the optimal model subset. The algorithm can also automatically predict the runtime of CSM application and select the optimal computing resources. To prove the advantages of the proposed approach, we evaluate different machine learning methods, such as linear regression, multilayer perceptron, regression tree and random forest. Experiments show that ours approach could effectively predict the runtime of CSM applications.

The proposed approach could be enhanced by the following future work:

(1) The generality of the proposed method can be considered to predict the runtime of different types of simulation applications.

(2) Explore more partition algorithms, expand the optional partition algorithm library and reduce the deployment cost of simulation applications.

ACKNOWLEDGEMENTS

This research was funded by the National Natural Science Foundation of China (no. 61903368 and no.61906207).

REFERENCES

Bocciarelli, P., D'Ambrogio, A., Mastromattei, A., Paglia, E., & Giglio, A. (2017). *Business process modeling and simulation: state of the art and MSaaS opportunities.* Paper presented at the Proceedings of the Summer Simulation Multi-Conference.

Cetinski, K., & Juric, M. B. (2015). AME-WPC: Advanced model for efficient workload prediction in the cloud. *Journal of Network & Computer Applications, 55*(sep.), 191-201.

Chen, T., He, T., & Benesty, M. (2016). xgboost: Extreme Gradient Boosting.

Cruz, R. M., Sabourin, R., Cavalcanti, G. D., & Ren, T. I. (2015). META-DES: A dynamic ensemble selection framework using meta-learning. *Pattern recognition, 48*(5), 1925-1935.

D'Angelo, G., & Marzolla, M. (2014). New trends in parallel and distributed simulation: From many-cores to cloud computing. *Simulation Modelling Practice and Theory, 49*, 320-335.

Dietterich, T. G. (2000). *Ensemble Methods in Machine Learning.* Paper presented at the International Workshop on Multiple Classifier Systems.

Fujimoto, R., Bock, C., Chen, W., Page, E., & Panchal, J. H. (2017). Research Challenges in Modeling and Simulation for Engineering Complex Systems.

Fujimoto, R. M. (2016). Research Challenges in Parallel and Distributed Simulation.

Fujimoto, R. M., Malik, A. W., & Park, A. (2010). Parallel and distributed simulation in the cloud. *SCS M&S Magazine, 3*, 1-10.

Haken, H. (2006). *Information and self-organization: A macroscopic approach to complex systems*: Springer Science & Business Media.

Hui, L., Groep, D., & Wolters, L. (2005). *An evaluation of learning and heuristic techniques for application run time predictions*.

Kaur, G., Bala, A., & Chana, I. (2019). An intelligent regressive ensemble approach for predicting resource usage in cloud computing. *Journal of Parallel and Distributed Computing, 123*, 1-12.

Kim, I. K., Wang, W., Qi, Y., & Humphrey, M. (2020). Forecasting Cloud Application Workloads with CloudInsight for Predictive Resource Management. *IEEE Transactions on Cloud Computing, PP*(99), 1-1.

Kurt, V., Silas, D. M., & Jan, B. (2012). *Conservative Distributed Discrete Event Simulation on Amazon EC2.* Paper presented at the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012).

Lee, B. D., & Schopf, J. M. (2004). *Run-time prediction of parallel applications on shared environments.* Paper presented at the IEEE International Conference on Cluster Computing.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM Computing Surveys (CSUR), 50*(6), 1-45.

Liu, X., Qiang, H., Qiu, X., Chen, B., & Huang, K. (2012). Cloud-based computer simulation: Towards planting existing simulation software into the cloud. *Simulation Modelling Practice & Theory, 26*(none), 135-150.

Matsunaga, A., & Fortes, J. A. (2010). *On the use of machine learning to predict the time and resources consumed by applications.* Paper presented at the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing.

Miu, T., & Missier, P. (2012). *Predicting the Execution Time of Workflow Activities Based on Their Input Features*. Paper presented at the Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis. https://doi.org/10.1109/SC.Companion.2012.21

Rahmanian, Asghar, A., Ghobaei-Arani, Mostafa, Tofighy, & Sajjad. (2018). A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment. *Future Generations Computer Systems Fgcs*.

Seneviratne, S., & Levy, D. C. (2011). Task profiling model for load profile prediction. *Future Generation Computer Systems, 27*(3), 245-255.

Tang, J., Alelyani, S., & Liu, H. (2014). Feature selection for classification: A review. *Data classification: Algorithms and applications*, 37.

Van der Laan, M. J., Polley, E. C., & Hubbard, A. E. (2007). Super learner.

Yang, J., Liu, C., Shang, Y., Cheng, B., Mao, Z., Liu, C., . . . Chen, J. (2014). A cost-aware auto-scaling approach using the workload prediction in service clouds. *Information Systems Frontiers, 16*(1), 7-18.

Yoginath, S. B., & Perumalla, K. S. (2015). Efficient Parallel Discrete Event Simulation on Cloud/Virtual Machine Platforms. *Acm Transactions on Modeling & Computer Simulation, 26*(1), 1-26.