Journal of Industrial Information Integration 27 (2022) 100337

Contents lists available at ScienceDirect

Journal of Industrial Information Integration

journal homepage: www.sciencedirect.com/journal/journal-of-industrial-information-integration

Workflow performance prediction based on graph structure aware deep attention neural network

Jixiang Yu [a,b,1], Ming Gao [c,d,*], Yuchan Li [c], Zehui Zhang [e], Wai Hung Ip [f,g], Kai Leung Yung [f]

[a] Microelectronics Thrust, Function Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangdong, China
[b] School of Data Science and Artificial Intelligence, Dongbei University of Finance and Economics, Dalian, China
[c] School of Management Science and Engineering, Key Laboratory of Big Data Management Optimization and Decision of Liaoning Province, Dongbei University of Finance and Economics, Dalian, China
[d] Center for Post-doctoral Studies of Computer Science, Northeastern University, Shenyang, China
[e] School of Economics and Management, Southwest Jiaotong University, Chengdu, China
[f] Department of Industrial and System Engineering, The Hong Kong Polytechnic University, Hong Kong SAR, China
[g] Department of Mechanical Engineering, University of Saskatchewan, Saskatoon, Canada

ARTICLE INFO

Keywords:
Workflow in cloud computing
Performance prediction
DAG structure
Deep Learning
DAG-Transformer

ABSTRACT

With the rapid growth of cloud computing, efficient operational optimization and resource scheduling of complex cloud business processes rely on real-time and accurate performance prediction. Previous research on cloud computing performance prediction focused on qualitative (heuristic rules), model-driven, or coarse-grained time-series prediction, which ignore the study of historical performance, resource allocation status and service sequence relationships of workflow services. There are even fewer studies on prediction for workflow graph data due to the lack of available public datasets. In this study, from Alibaba Cloud's Cluster-trace-v2018, we extract nearly one billion offline task instance records into a new dataset, which contains approximately one million workflows and their corresponding directed acyclic graph (DAG) matrices. We propose a novel workflow performance prediction model (DAG-Transformer) to address the aforementioned challenges. In DAG-Transformer, we design a customized position encoding matrix and an attention mask for workflows, which can make full use of workflow sequential and graph relations to improve the embedding representation and perception ability of the deep neural network. The experiments validate the necessity of integrating graph-structure information in workflow prediction. Compared with mainstream deep learning (DL) methods and several classic machine learning (ML) algorithms, the accuracy of DAG-Transformer is the highest. DAG-Transformer can achieve 85-92% CPU prediction accuracy and 94-98% memory prediction accuracy, while maintaining high efficiency and low overheads. This study establishes a new paradigm and baseline for workflow performance prediction and provides a new way for facilitating workflow scheduling.

1. Introduction

With the rapid development of the Internet of Things, big data, and e-commerce, the needs for versatile and elastic cloud computing infrastructures have surged. Top cloud service vendors, such as Amazon AWS, Microsoft Azure, and Alibaba Cloud have launched a series of cloud computing technologies and business models. More edge computing devices are also deployed in process-intensive scenarios, such as manufacturing and industrial control. Recent years, container technology such as docker and Kubernetes has become the dominated virtualized resource management mechanism for its lightweight, easy-to-deploy, high-efficiency, and resource-sharing features [1]. The container technology is capable of high-density deployment and real-time elastic scaling of cloud services. Elastic scheduling adjusts resources supply according to the future tasks' workload, which can reduce the resource waste during idle periods and complement the resources demand during busy periods. However, it creates great complexity in management and optimization decisions. Therefore, a considerable amount of research is focused on task scheduling models and related optimization algorithms [2,3,4,5].

* Corresponding author.
E-mail address: gm@dufe.edu.cn (M. Gao).
[1] Jixiang Yu will join The Hong Kong University of Science and Technology (Guangzhou) as a PhD student in Fall, 2022

https://doi.org/10.1016/j.jii.2022.100337

Available online 9 February 2022

In elastic resource scheduling, the estimation of the resource demand of unexecuted tasks is a critical effort. However, traditional resource scheduling utilizes lab measurements, manual experience, and simple statistics [6] to adjust the resource allocation accordingly. They do not consider historical logs and future trends of resource usage. In complex scenarios, the resource mismatches often occur between demand and supply, which result in low overall resource utilization. Accurate predictions are placed high hopes to provide precise scheduling, which is gradually considered as a new way to improve cloud utilization and reducing energy consumption [7, 8, 9, 10, 11].

In fact, performance prediction in clouds is a classical problem, such as resource occupation or computational workload. But previous research is mainly model-driven [12, 13, 14, 15], and based on a single virtual machine (VM)/instance, single task, or single host. However, there are relatively few research on performance predictions for tasks organized as workflows, especially in the new generation of containerized clouds. Such performance predictions can help cloud providers optimize business processes in an end-to-end manner for various deployment modes in hybrid clouds. Thus, there is an urgent need for prediction models that provide accurate performance estimation of future tasks in workflows.

To this end, we first extract, clean and transform containerized workflow logs from Cluster-trace-v2018 of the Alibaba Cloud. We aggregate nearly one billion offline task instance records into a new dataset, which contains approximately one million workflows, and extract their corresponding structural information into DAG matrices.

To make use of these fine-grained graph spatiotemporal sequences, which represents historical performance, resource allocation, and service relationships of workflows, this study proposes an improved Transformer method (i.e., DAG-Transformer). According to the workflow's definition, a novel position encoding suitable for DAG is proposed to decorate workflow sequence, in which the parallel relationship of tasks can be well represented. Simultaneously, a corresponding attention mask is introduced in the attention calculation phase of DAG-Transformer, which can regularize attention calculations according to the dependencies between workflow tasks. Hence, DAG-Transformer embeds sequence and graph relationships of workflows together to enhance the feature representation of the deep neural network.

DAG-Transformer can predict the performance of upcoming tasks based on the historical performance of the workflows, providing key parameters to the workflow scheduling algorithms. We find that the effective use of DAG structure information consistently improves the performance of our proposed prediction model, achieving an overall prediction accuracy of 85-92% for CPU and of 94-98% for memory. Our proposed model and exploration, for the first time, emphasize the importance of prediction-driven scheduling in cloud workflows from a quantitative perspective, giving a clear, intuitive and detailed baseline.

Fig. ;1 shows the framework of this research.

2. Literature Review

2.1. The necessity of predictions for cloud scheduling

Cloud computing platforms need to allocate computing resources reasonably when providing services, so that tasks submitted by users can be executed in a shorter time and at a lower cost [16]. However, with the rapid increase of users and applications, the cloud environment has become more complex than ever, and the problem of low resource utilization in the cloud environment has become very prominent [6]. According to the comparison in [17], the average workload fluctuation of a cloud environment is approximately 20 times larger than that in a traditional grid.

Traditionally, by allocating resources based on a fixed threshold, cloud providers are hard to achieve consistent Quality of Service (QoS) while saving energy and cost. In academia, a wide range of scheduling algorithms are proposed to manage the cloud. They treat effective and efficient scheduling algorithms as a solution to the abovementioned problem [16, 18, 19, 20, 21, 22]. However, underneath the scheduling, accurate performance prediction is a prerequisite for correct decision-making. Yang et al. [7], Shaw et al. [8], Zhong et al. [9], Aslam et al. [10], and Zhu et al. [11], all illustrated the significance of predictions for scheduling, optimization, load balancing, energy savings, etc. Duggan et al. [23] considered that prediction algorithms can provide a greater chance to prevent a host from becoming overutilized when sudden high demands occur. Kim et al. [24] believed that predictive resource management is highly dependent on workload predictors, which can estimate short/long term fluctuations in cloud application workloads, but current methods cannot deal with all cloud workload patterns. There are also some related works [23, 25, 26, 27, 28, 29, 30, 31, 32, 33] that use prediction results to provide a basis for better scheduling algorithms. We organized the Table 1 to show the previous works related to performance predictions in a cloud environment. From the literature, we find that performance prediction, which is a vital foundation of optimization and scheduling in cloud environments, can help vendors reduce the complexity of cloud management. In complex hybrid cloud environments, where fluctuating resource demands lead to low overall resource utilization, predictions can provide precise parameters for scheduling, thereby improving cloud utilization and reducing energy consumption. Predictions play an increasingly important role and have become a hot topic of recent research.

2.2. Cloud workflows

For complex cloud services, offline batched tasks can often be orchestrated as workflows based on DAG structure, which involves parallelism and dependencies between the workflow's affiliated tasks [45]. Furtherly, a task is considered as completed only if its underlying instances are all executed successfully. This is shown in Fig. ;2.

Existing approaches have proposed the importance of prediction for scheduling and integrated the prediction step into scheduling algorithms. However, limited by the scarcity of available datasets, prediction

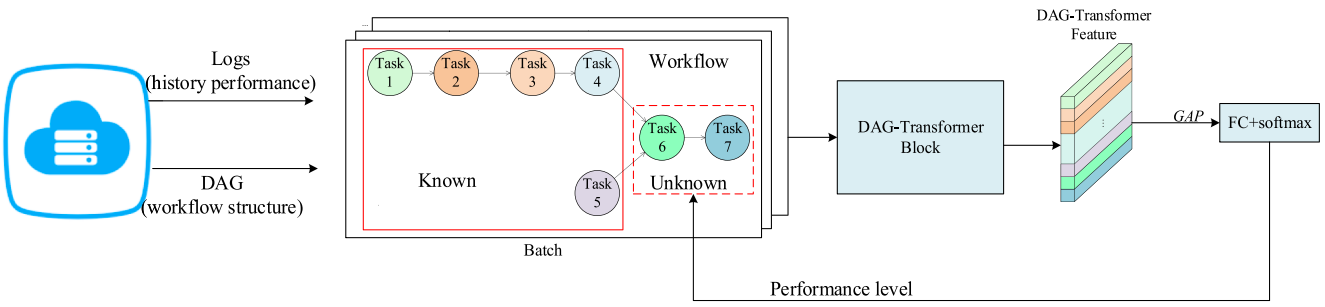


Fig. 1. Research framework.

Table 1
Previous works related to cloud predictions.

	Year	Prediction index	Method used	Whether to schedule based on prediction results	Datasets or Simulator	Dataset(s) used
Jiang et al. [34]	2013	VM capacity	Ensemble learning	No	Dataset	IBM Smart Cloud Enterprise
Yang et al. [7]	2015	CPU workload	ESN based autoencoder	No	Dataset	Google Cluster 2011
Shaw et al. [8]	2017	Bandwidth availability	ARIMA	Yes	Simulator	-
Janardhanan et al. [35]	2017	CPU workload	ARIMA and LSTM	No	Dataset	Google Cluster 2011
Zhong et al. [9]	2018	CPU workload	Weighted wavelet SVM	No	Dataset	Google Cluster 2011
Duggan et al. [23]	2018	Bandwidth usage and CPU workload	RNN	Yes	Dataset+simulator	Google Cluster 2011 and Amazon EC2
Gupta et al. [36]	2018	CPU usage	BiLSTM	No	Dataset	Google Cluster 2011
Zhang et al. [37]	2018	CPU Utilization	Autoencoder	No	Dataset	PlanetLab
Aslam et al. [10]	2019	CPU and Memory usage, Disk I/O time	heuristics	No	Dataset	Google Cluster 2011
Zhu et al. [11]	2019	CPU workload	LSTM based encoder-decoder	No	Dataset	Alibaba cluster-trace-v2018 and Dinda(single machine)
Erradi et al. [38]	2019	CPU, memory, bandwidth utilization	MLP, Linear Regression	No	Simulator	-
Fei et al. [39]	2020	Number of tasks in a cluster	ARIMA	Yes	Dataset+Simulator	Google Cluster 2011
Gao et al. [40]	2020	CPU and Memory usage	LSTM, ARIMA	No	Dataset	Google Cluster 2011
Suksriupatham et al. [26]	2020	CPU Utilization	SVM/Linear Regression/ Regression Tree/ polynomial regression	Yes	Dataset+Simulator	PlanetLab
Hsieh et al. [27]	2020	CPU Usage	Gray–Markov	Yes	Dataset+Simulator	PlanetLab
Xiao et al. [28]	2020	VM requirements	heuristics	Yes	Dataset + Simulator	Wikinews
Marahatta et al. [29]	2020	task failure probability	MLP	Yes	Dataset + Simulator	Eular Data Set and Internet Data Set
Rjoub et al. [41]	2020	-	DRL(LSTM)	-	Dataset+Simulator	Google Cluster 2011
Li et al. [30]	2020	HTTP requests	DBN	Yes	Dataset + physical environment	Baidu Network Trafc Statistics institute
Kim et al. [24]	2020	Incoming jobs, user requests, workload	Ensemble learning	No	Dataset+Simulator	Google Cluster 2011, Facebook Hadoop; Wikipedia web traces; Grid Workloads Archive
Kholidy [42]	2020	CPU utilization, Disk IOs/sec, memory utilization, and Network bandwidth used	PSO,ARIMA,SVM	No	Simulator	-
Tan et al. [16]	2021	-	DQN+PSO	-	Dataset	QWS and WSDream
Prassanna et al. [25]	2021	workload state	heuristics	Yes	Dataset + Simulator	Amazon EC2
Davami et al. [31]	2021	Maximum tasks to execute in parallel	LSTM	Yes	Dataset+Simulator	DS Lab
Bi et al. [43]	2021	Workload, CPU usage and Memory usage	Combination of Bi-LSTM and Grid-LSTM	No	Dataset	Google Cluster 2011
Karim et al. [44]	2021	CPU workload	Combination of LSTM and CNN	No	Dataset	GRID Workloads Archive
Kaur et al. [32]	2021	availability of the resources	Ensemble learning	Yes	Simulator	-
Yeung et al. [33]	2021	GPU Utilization & VRAM Occupation	heuristics	Yes	Dataset + Simulator	Traces in Tiresias

models specific for workflows have not been studied systematically and applied in scheduling.

Previous cloud predictions mainly focus on a single VM/host/cluster/task, and most of them are modeled as time-series, using (*t-n*….*t-1*) to predict *t* [8, 35, 39]. They cannot effectively make use of the dependency relationships of tasks, and each task is isolated from each other. Wu et al. [45] stated that workflow has become a paradigm to explain tasks on the cloud, but there are still few performance predictions facing cloud workflows. Kousalya et al. [46] proposed two prediction tracks for workloads of cloud workflows, one is simulation and test environment-based measurement, and the other is log-based data-driven machine learning. They elaborate on the necessity and feasibility of workflow-oriented scheduling in the cloud, especially the data-driven way. However, there is no mention of how to use contextual and structural features, and no comparison for ML-based prediction methods.

The logs and definition of workflow can reflect historical performance, resource allocation, and service relationship. As Wu et al. [45] proposed, the performance predictions towards cloud workflows are scarce. One of the main reasons is the lack of a dataset. In our prior work [47], we investigated related public cluster datasets released by cloud computing vendors, as shown in Table 2, among which, Cluster-trace-v2018 released by the Alibaba Cloud appears to be an appropriate research object.

2.3. Challenges in the Alibaba Cloud

Among the public cluster traces, only ClusterData2019 released by Google [48] and Cluster-trace-v2018 released by Alibaba Cloud [49] contain DAG information. Considering the data accessibility, we select Cluster-trace-v2018 for research. There are several previous works that have explored this dataset, including analysis of resource efficiency and utilization [51, 52], characterization of task dependencies [72], and exploration of co-located services [6]. Liu et al. [50] and Guo et al. [51] both noted that the planned resources requirements are highly inconsistent with that actually used. The problems of over-allocation and under-allocation of resources are both significant.

Liu et al. [50] concluded that batch instances with low resource requests tend to overcommit both CPU and memory at runtime. In the contrast, there are also batch instances that underutilize the resources they request. In [50], it can be seen that the actual resource usage at runtime may exceed the request by more than 10 times, and this pattern

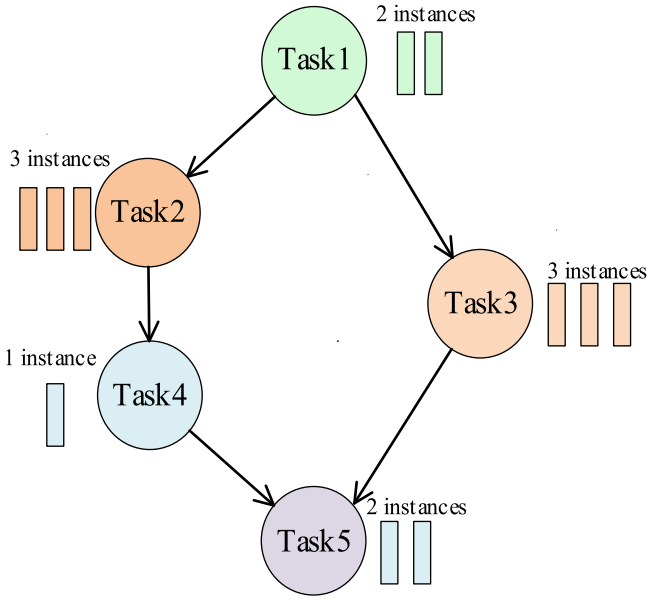


Fig. 2. An example of workflow-task-instance DAG

varies from different resources (e.g. CPU and memory). Guo et al. [51] also reported that most of the highest average utilizations may exceed 100% of the resources requested in the Alibaba Cloud.

Another work from the Alibaba Cloud [6] noted that online services are latency-critical but may not consume much CPU and/or memory resources. On the other hand, offline batch workflows demand as many resources as possible to maximize their performance. This type of co-location mode leads to the resource imbalance in Alibaba Cloud. Under such circumstances, the performance response model constructed from dedicated test bed in lab is subject to deviation when the workload interference of online tasks, user interaction, etc. exist [53].

In conclusion, it has been noted in related data analysis and qualitative studies that the resource demand planned by workflow designers in ideal experimental environments often deviates from the workloads in real production environments, especially under the online-offline co-located hybrid containerized deployment mode. Although load interference estimation can address the above issues, it is highly complex and almost infeasible [6]. As above, it becomes an intuitive idea to establish a data-driven and machine learning based method for performance prediction in the new generation of high-density, green, hybrid heterogeneous clouds.

Additionally, according to Lu et al. [54], in Cluster-trace-v2018, the number of workflows with different numbers of tasks varies greatly. After preliminary statistics (please see the appendix), we see that among workflows with the same number of tasks, their structures also vary. For example, in workflows with 3 tasks, there are as many as 30 different workflow structures, and there are as many as 1261 different workflow structures in workflows with 7 tasks. Due to the high complexity of cloud workflows, traditional mathematical models, heuristics-based algorithms, and traditional machine learning methods have limitations in modeling different workflow structures. They cannot identify and infer implicit relationships among individual tasks without graph structure-aware computation. As a result, they would lose some useful information which may improve the prediction accuracy.

2.4. Prospects and potential of DL

The end-to-end representation learning methods represented by DL have achieved inspiring success in various fields, such as computer vision [55], speech recognition [56], and natural language processing [57]. Recent progress in DL refreshes people's recognition of artificial intelligence. OpenAI trains a language model, GPT-3 [58], with 175 billion parameters, allowing intelligent chatbots to become possible. AlphaFold [59] proposed by DeepMind can accurately predict protein structures at the atomic level, even when similar structures are not known. Moreover, algorithms based on DL can even nowcast precipitation skillfully [60]. In particular, the Transformer architecture [61] and self-attention mechanism demonstrate superior potential for different fields [62, 63, 64]. Previous works in Table 1 have shown the feasibility of applying deep learning to cloud predictions, but the application of Transformer is scarce. To this end, we are inspired to explore deep learning methods and Transformer-based architectures to address workflow performance predictions.

Many current Transformer-based DL algorithms are designed for specific areas, such as natural language processing (Bert [62]) and computer vision (ViT [63]). However, workflows are essentially graphs, whose data structure are different from natural language and images (sequences and patches). These Transformer-based methods cannot elegantly and exactly model the DAG information. As a consequence, they cannot be migrated to cloud workflows directly. To address this issue, we design a DAG-oriented Transformer variant to adapt to the cloud workflow.

2.5. Our motivation

Our motivation comes from observation of the Cluster-trace-v2018 and insights of the prior works related to this trace [6, 50, 51, 52, 53, 54]. We find that in real production systems, the planned resources are highly mismatched with the actual resource demand, causing under-utilization and overutilization phenomena. Efficient scheduling to improve resource utilization in the cloud relies on the accuracy of the estimation of future indicators. The importance of better perception from a global perspective is also mentioned in cloud community [51]. Hence, we aim to provide accurate, real-time estimations of such indicators from a workflow-level perspective.

The challenges of workflow-oriented predictions are as follows:

1 How to cope with the diverse workflow structures.
2 How to convert cloud traces (graphs) into ML-trainable samples.
3 How to design a suitable DL algorithm (e.g., Transformer) for cloud workflow predictions.

Overall, we focus on how to utilize a representative large-scale cloud trace (i.e., Cluster-trace-v2018) to form a workflow graph dataset, establish a workflow-oriented prediction model, and provide a strong

Table 2
Public cloud cluster traces

Cloud Vendor	Dataset	Released Year	Period	Content
Google	ClusterData2011	2011	1 month	A single 12.5 k-machine Borg cell
	ClusterData2019	2019	1 month	Eight Borg cells
Azure	AzurePublicDatasetV1	2017		~2 M VMs and 1.2 B utilization readings
	AzurePublicDatasetV2	2019		~2.6 M VMs and 1.9 B utilization readings
	AzureFunctionsDataset2019	2019	2 weeks	A subset of applications running on Azure Functions
Alibaba Co-located cluster	Cluster-trace-v2017	2017	12 hours	About 1300 machines
	Cluster-trace-v2018	2018	8 days	About 4000 machines, the DAG information of production batch workloads

baseline. In addition, the proposed approach is applicable to similar workflow management other than cloud computing communities, such as smart manufacturing, logistics and supply chain, transportation, and IoT industries.

3. Methodology

3.1. Descriptive exploration of Alibaba Cluster

At the end of 2018, Alibaba released Cluster-trace-v2018. This dataset contains the operational logs of co-located online and offline container services on 4,023 servers in 8 days. We analyze the CPU utilization in this dataset, as shown in Fig. ;3. Although approximately 90% of servers have a maximum utilization of CPU above 80%, more than 80% of the server's average CPU utilization is between 30% and 50%. The difference between the maximum utilization and the average utilization is more than 40%.

Comparing the average and maximum utilization of CPU, we see that although the maximum utilization of server resources reaches a relatively high level in certain periods, the average utilization of resources is still at a middle level, which shows that the clusters still have lower overall utilization of resources. The phenomenon of low resource utilization is possibly due to the unreasonable scheduling lack of accurate prediction for actual resource usage of services.

3.2. Preprocessing of the dataset

Two logs of offline batch workflows are recorded in Cluster-trace-v2018. One log is the "batch_task" table, and the other log is the "batch_instance" table.

Log 1 is the "batch_task" table, which describes the tasks' performance in the batch workflows, as shown in Table 3.

Log 2 is the "batch_instance" table, which describes the performance of instances which constitute tasks in the batch workflows, as shown in Table 4.

The above datasets don't provide the graph structure of each workflow and task-level performance, so we design the preprocessing pipeline. The steps to extract the workflow are shown in Fig. ;4. ❶: First, we derive the DAG information based on the name field of the task, and group the tasks according to the offline workflow to which they belong. ❷: Then, we filter out all instances of the specific workflow from the "batch_instance" table and aggregate the instance performance to form the task-level performance.

We finally exact 4,201,013 batch workflows. We find that a workflow can have as few as 1 task and as many as 1,002 tasks. Workflows with fewer than 10 tasks (3,980,466) account for 94.75% of the total workflows, and workflows with fewer than 20 tasks (4,171,498) account for 99.30% of the total workflows. We see that overly complex workflows exist, but they are not common. We also find that the structure of batch workflows with the same number of tasks varies, as shown in Fig. ;5.

Table 3
"batch_task (workflow task)" table.

Feature Name	Interpretation
tid	Id of task
inst_num	Number of instances included in the task
task_type	Type of task
jid	The name of the workflow to which the task belongs
status	Status of task
stime	Start time of task
etime	End time of task
plan_cpu	CPU requested for each instance of the task
plan_mem	Memory requested for each instance of the task

Table 4
"batch_instance (workflow instance)" table.

Feature Name	Interpretation
ins_nam	Name of instance
tid	The name of the task to which the instance belongs
jid	The name of the workflow to which the instance belongs
stime	Start time of instance
etime	End time of instance
mid	The name of the machine to which the instance belongs
cpu_avg	The average CPU usage of the instance
cpu_max	The max CPU usage of the instance
mem_avg	The average memory usage of the instance
mem_max	The max memory usage of the instance

The preprocessing results of the workflows with 3 to 50 tasks are listed in Table 5. Figures about the analysis of these workflows are shown in appendices.

❶: The DAG information is transformed into a sparse matrix, and the connection weights between tasks are described through in-degree and out-degree, as shown in Figure 6 and Figure 7. "o_" represents an out-degree, and "i_" represents an in-degree. For task 6, its "o_7" value is 1, which means that task 7 starts after the end of task 6. The values of "i_4" and "i_5" are 0.5, which means that the precondition for the start of task 6 is the end of task 4 and task 5.

❷: We calculate the average, variance, maximum, minimum, median, skewness and kurtosis of the instances' resource occupation (CPU, memory) of each task in the workflow, as well as the average, variance, maximum, minimum of the tasks' execution time (lifecycle), to construct the task-level performance.

The features of a task's performance and DAG information in workflow are shown in the tables in appendices.

For the convenience and without loss of generality, we conduct the

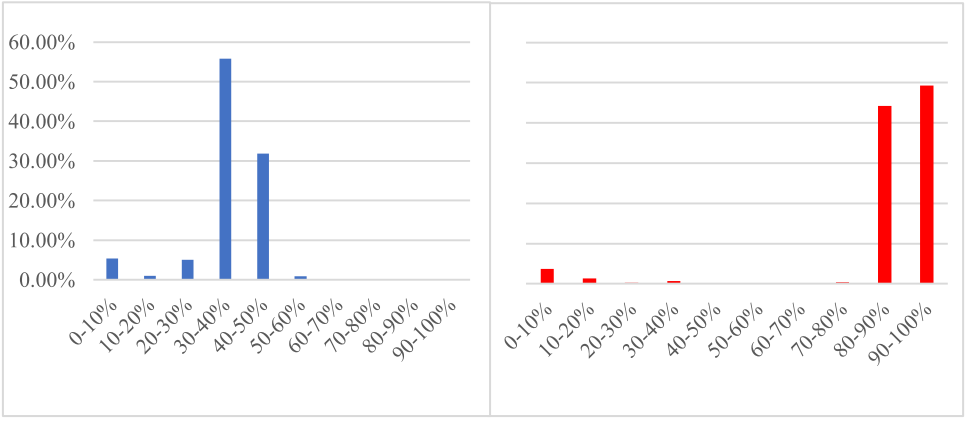
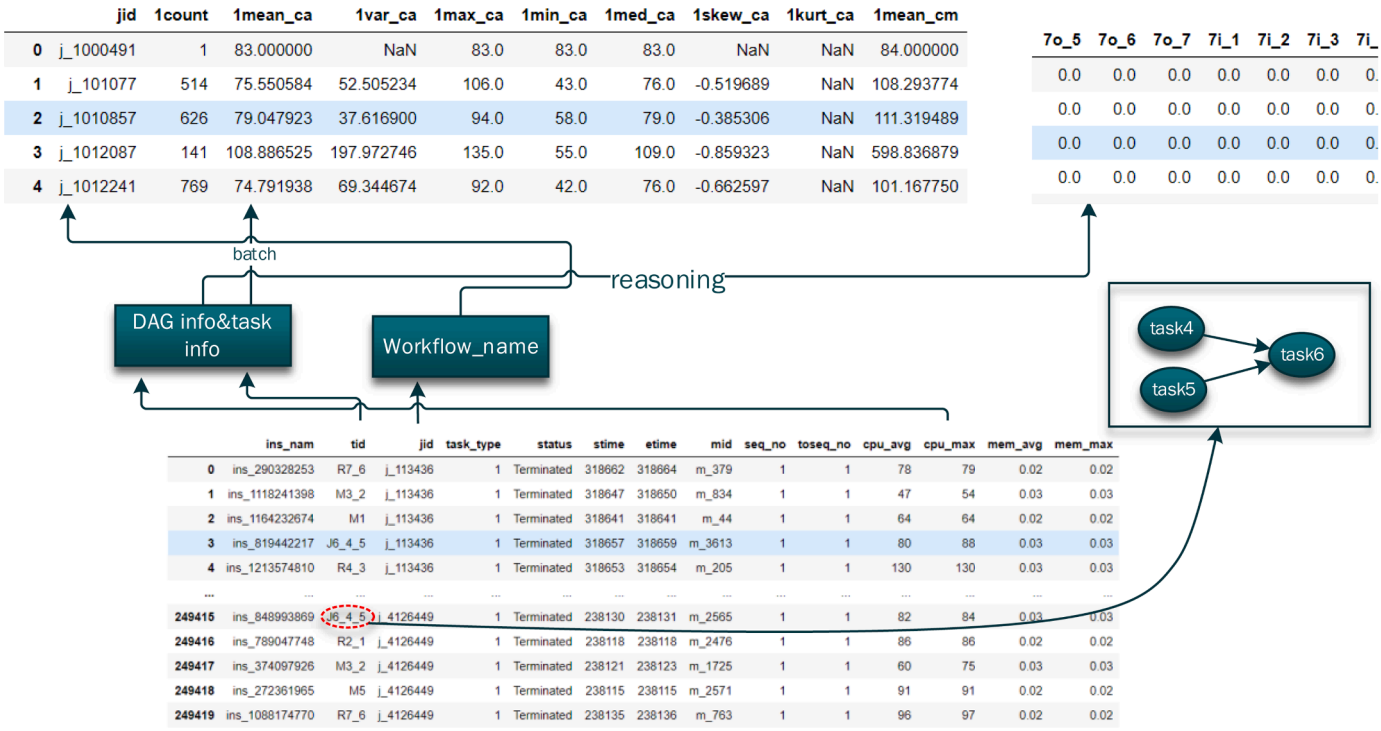


Fig. 3. Average (Blue, left) and Maximum (Red, right) of CPU utilization of servers in Cluster-trace-v2018



	jid	1count	1mean_ca	1var_ca	1max_ca	1min_ca	1med_ca	1skew_ca	1kurt_ca	1mean_cm
0	j_1000491	1	83.000000	NaN	83.0	83.0	83.0	NaN	NaN	84.000000
1	j_101077	514	75.550584	52.505234	106.0	43.0	76.0	-0.519689	NaN	108.293774
2	j_1010857	626	79.047923	37.616900	94.0	58.0	79.0	-0.385306	NaN	111.319489
3	j_1012087	141	108.886525	197.972746	135.0	55.0	109.0	-0.859323	NaN	598.836879
4	j_1012241	769	74.791938	69.344674	92.0	42.0	76.0	-0.662597	NaN	101.167750

7o_5	7o_6	7o_7	7i_1	7i_2	7i_3	7i_
0.0	0.0	0.0	0.0	0.0	0.0	0.
0.0	0.0	0.0	0.0	0.0	0.0	0.
0.0	0.0	0.0	0.0	0.0	0.0	0.
0.0	0.0	0.0	0.0	0.0	0.0	0.

	ins_nam	tid	jid	task_type	status	stime	etime	mid	seq_no	toseq_no	cpu_avg	cpu_max	mem_avg	mem_max
0	ins_290328253	R7_6	j_113436	1	Terminated	318662	318664	m_379	1	1	78	79	0.02	0.02
1	ins_1118241398	M3_2	j_113436	1	Terminated	318647	318650	m_834	1	1	47	54	0.03	0.03
2	ins_1164232674	M1	j_113436	1	Terminated	318641	318641	m_44	1	1	64	64	0.02	0.02
3	ins_819442217	J6_4_5	j_113436	1	Terminated	318657	318659	m_3613	1	1	80	88	0.03	0.03
4	ins_1213574810	R4_3	j_113436	1	Terminated	318653	318654	m_205	1	1	130	130	0.03	0.03
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
249415	ins_848993869	J6_4_5	j_4126449	1	Terminated	238130	238131	m_2565	1	1	82	84	0.03	0.03
249416	ins_789047748	R2_1	j_4126449	1	Terminated	238118	238118	m_2476	1	1	86	86	0.02	0.02
249417	ins_374097926	M3_2	j_4126449	1	Terminated	238121	238123	m_1725	1	1	60	75	0.03	0.03
249418	ins_272361965	M5	j_4126449	1	Terminated	238115	238115	m_2571	1	1	91	91	0.02	0.02
249419	ins_1088174770	R7_6	j_4126449	1	Terminated	238135	238136	m_763	1	1	96	97	0.02	0.02

Fig. 4. Pipeline to extract the workflow

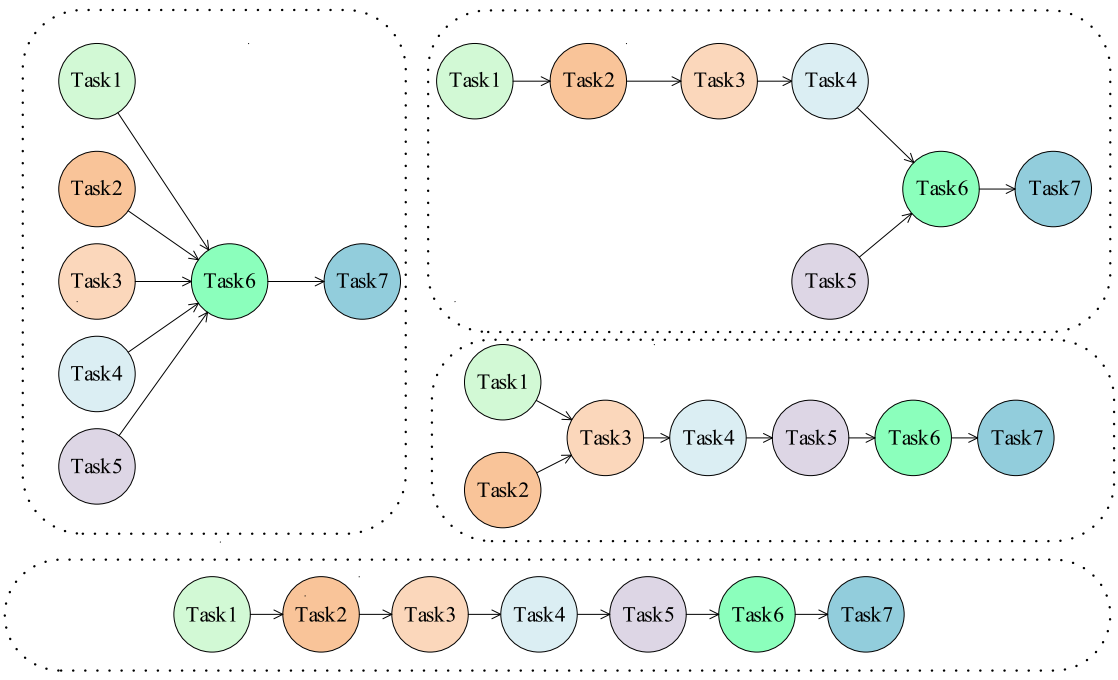


Fig. 5. Examples of workflows' different structures

Table 5
Preprocessing results of workflows with 3 to 50 tasks

Statistics basis	total
Number of instances	1,069,255,469
Number of tasks	8,108,011
Number of workflow samples	1,208,692
Number of workflow structures	14,972

prediction study on batch workflows with 7 tasks due to their sufficient samples and diversity of structures.

3.3. Generation of labels

From the perspective of container resource allocation, due to the complexity of billing and scheduling algorithms, the configuration of cloud servers is often based on a limited combination of predefined container configurations, such as CPU cores(2,4,8,…) and memory

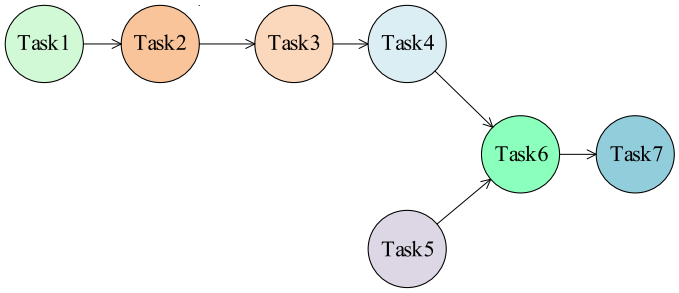


Fig. 6. An example of workflow with 7 tasks

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	i_1	i_2	i_3	i_4	i_5	i_6	i_7
Task1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Task2	0	0	1	0	0	0	0	1	0	0	0	0	0	0
Task3	0	0	0	1	0	0	0	0	1	0	0	0	0	0
Task4	0	0	0	0	0	1	0	0	0	1	0	0	0	0
Task5	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Task6	0	0	0	0	0	0	1	0	0	0	0.5	0.5	0	0
Task7	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fig. 7. DAG information matrix of the workflow shown in Fig. 6

capacity(4,8,16,...). Therefore, interval prediction can basically satisfy scheduling needs, and we define the prediction problem as a classification model. Fig. ;8 shows the generation steps of classification labels according to the tasks' performance.

We use K-means clustering to divide the tasks' performance into three levels: low, medium, and high, denoted as 0, 1, and 2 in the labels [65, 67]. Through quantitative and qualitative evaluation, we choose features "mean_ca" and "max_ca" for clustering, which is a more discriminative method for reflecting the performance level of the task. Fig. ;9 shows the t-SNE visualization of task 7's CPU performance [66].

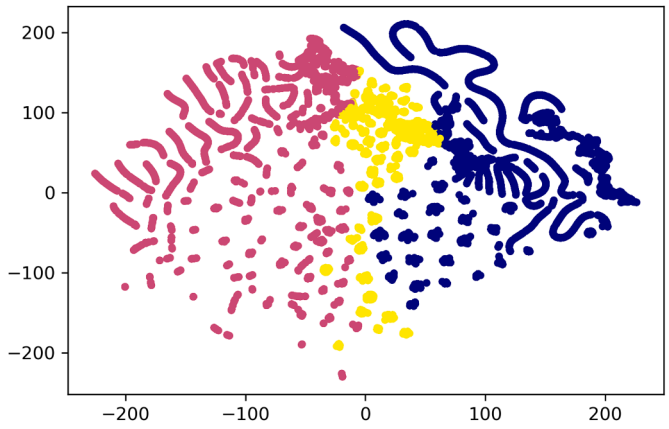


Fig. 9. t-SNE visualization

3.4. DAG-Transformer

We customize the Transformer Encoder to make it suitable for graph structured data learning, and we also design the corresponding position encoding and attention mask for the workflows. Fig. ;10 shows a complete overview of the model architecture.

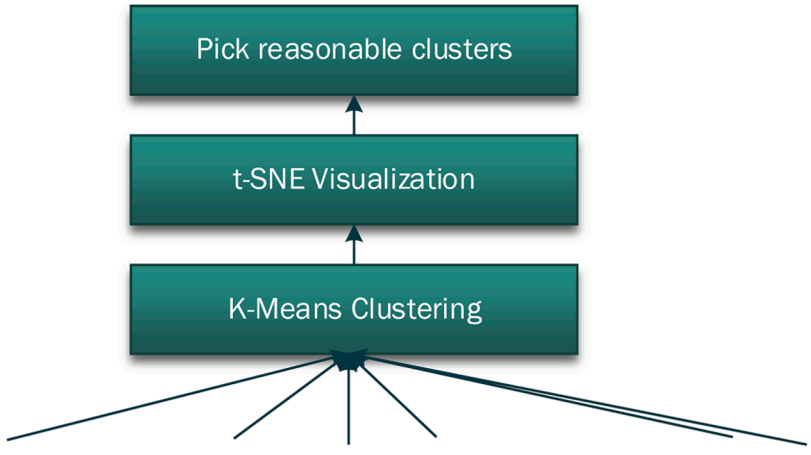
3.4.1. Extraction of Tasks' Performance and DAG information

We first perform a min-max scaling on all task features, which is shown in the equation below.

$$t_i = \frac{t_i - \min(t_i)}{\max(t_i) - \min(t_i)},$$

where $\max(t_i)$ and $\min(t_i)$ are the maximum and minimum values of all the tasks' i-th feature in the dataset respectively. Then, the normalized workflow data X can be represented as $X = [T_1, T_2, T_3, ..., T_n]$, where T_i denotes the i-th task's features in the workflow. X is then fed into the network as input which is followed by training and evaluation.

The DAG information shown in Fig. ;7 can be split into weighted out-



	jid	1count	1mean_ca	1var_ca	1max_ca	1min_ca	1med_ca	1skew_ca	1kurt_ca	1mean_cm	...
0	j_1000491	1	83.000000	NaN	83.0	83.0	83.0	NaN	NaN	84.000000	...
1	j_101077	514	75.550584	52.505234	106.0	43.0	76.0	-0.519689	NaN	108.293774	...
2	j_1010857	626	79.047923	37.616900	94.0	58.0	79.0	-0.385306	NaN	111.319489	...
3	j_1012087	141	108.886525	197.972746	135.0	55.0	109.0	-0.859323	NaN	598.836879	...
4	j_1012241	769	74.791938	69.344674	92.0	42.0	76.0	-0.662597	NaN	101.167750	...

Fig. 8. Steps to label the performance

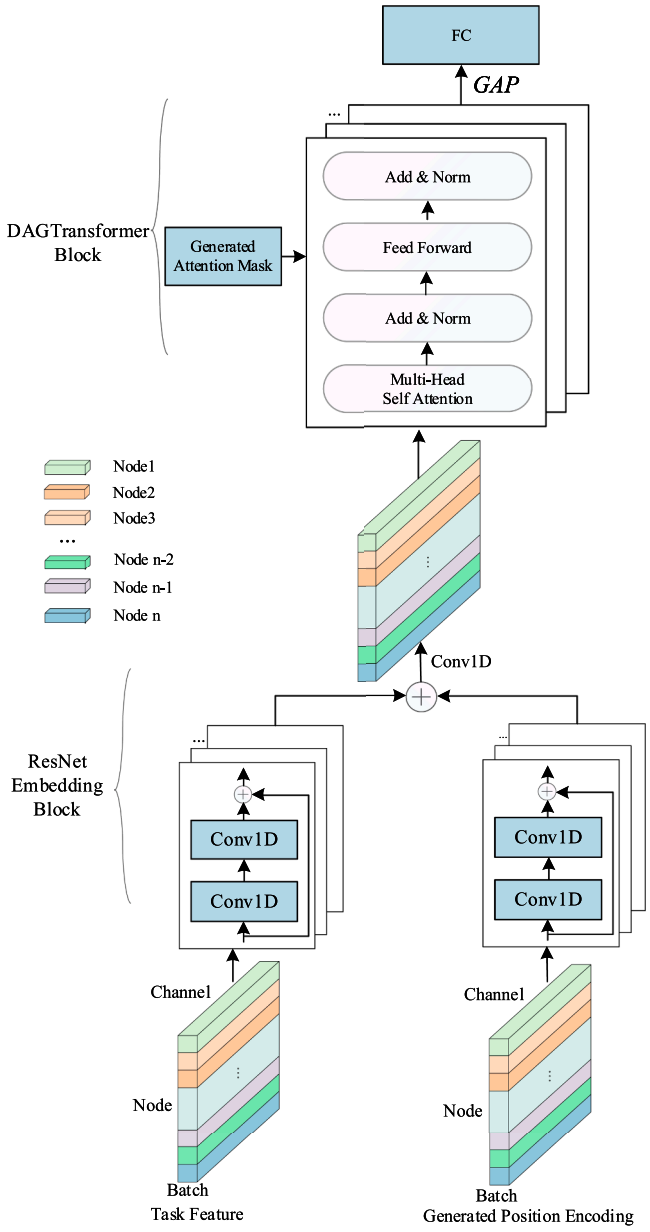


Fig. 10. An overview of the DAG-Transformer

degree matrix A_1 and weighted in-degree matrix A_2. These two matrices are subsequently used to generate position encoding and attention masks.

3.4.2. Design of position encoding

For the Transformer and self-attention mechanism, position encoding is quite important. If there is no position encoding added in sequence, each node will be treated as positional equivalent, unrelated, and isolated component, which is similar to classic tabular machine learning methods.

Aiming at the precise representation of the specific graph structure of the workflow, we design a position encoding approach based on the dependency relationship of the nodes in the graph. The algorithm of generating position encoding is improved on the basis of the Bellman-Ford algorithm and is shown in **Algorithm 1**.

As the example shown in Fig. ;11 below, the generated vector p is [0,1,2,3,0,4,5]. Subsequently, p is used to generate position encoding matrix P, which is also fed into the network as input.

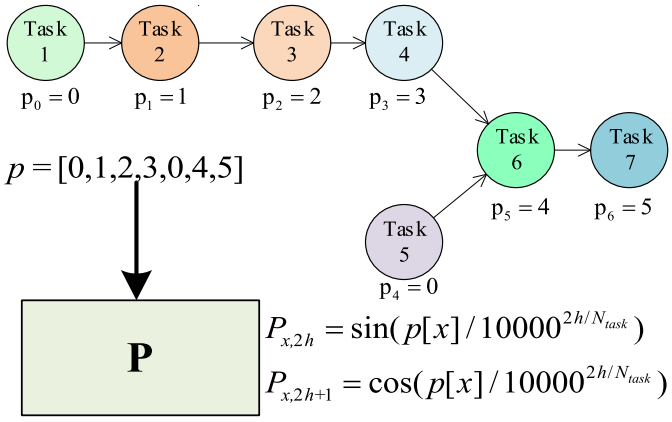


Fig. 11. An intuitive example of position encoding

3.4.3. Model architecture

3.4.3.1. Embedding block. In our model, the position encoding matrix P and workflow data matrix X are fed into separate customized ResNet blocks. The ResNet [68] selectively extracts features from different layers (with shortcut connections) and combines them with the original features, which has been extensively verified empirically, especially in very deep networks to alleviate the problem of gradient disappearance. As shown in Fig. ;12, in our customized ResNet block, we remove the ReLU activation function to retain more linear information, and postpone the non-linear interaction in the following attention module. The calculation in customized ResNet layer can be represented as:

$$\mathrm{ResNet}(X) = X + \mathrm{Conv1D}(\mathrm{Conv1D}(X)).$$

Therefore, the calculations in initial embedding phase are:

$$\begin{aligned} X &= \mathrm{ResNet}_1^{(l)}(X) \\ P &= \mathrm{ResNet}_2^{(l)}(P) \end{aligned},$$

where l denotes the number of layers of ResNet.

Then, X and P are added and fed into another Conv1D layer, which can be represented as:

$$\begin{aligned} &X \in \mathrm{R}^{N_{task} \times d_k} \\ &= \mathrm{Conv1D}(X + P), \end{aligned}$$

where N_{task} denotes the number of tasks in this workflow, and d_k denotes the embedding dimension.

This intermediate result X is subsequently fed into Transformer Encoder.

3.4.3.2. DAG-Transformer Encoder. We generate an attention mask according to the workflow's DAG structure. The algorithm used to generate the attention mask is shown in **Algorithm 2**.

In the self-attention calculation phase of Transformer Encoder, Q, K, and V all equal to the input (i.e., intermediate result $X \in \mathrm{R}^{N_{task} \times d_k}$ in *3.4.3.1*). The self-attention calculation in the vanilla Transformer can be expressed as:

$$attn = \mathrm{self-attention}(X) = \mathrm{softmax}\left(\frac{(W_Q Q)(W_K K)^{\mathrm{T}}}{\sqrt{d_k}}\right)(W_V V),$$

where W_Q, W_K, W_V are weights of linear projection, and d_k denotes the embedding dimension. In our DAG-Transformer Encoder, the attention mask is applied after the dot product of Q and K, which can be expressed as:

$$attn = \mathrm{self-attention}(X) = \mathrm{softmax}\left(\frac{(W_Q Q)(W_K K)^{\mathrm{T}}}{\sqrt{d_k}} + M\right)(W_V V).$$

After the dot product of $(W_Q Q)$ and $(W_K K)$, the dimension is converted from $\mathrm{R}^{N_{task} \times d_k}$ to $\mathrm{R}^{N_{task} \times N_{task}}$, and this product is the attention of each task node to any task nodes in the workflow. If task T_i is connected to task T_j, or if there is a dependency relationship between T_i and T_j, then 0 is added to the i-th row, j-th column and j-th row, i-th column of $(W_Q Q)(W_K K)^{\mathrm{T}}$. Otherwise, -inf is added to the i-th row, j-th column and j-th row, i-th column. Due to the calculation method of the softmax activation function, the attention mask assigns the unconnected nodes an extremely lower weight, and almost covers their attention. The Fig. ;13 shows an intuitive example of how the attention mask works.

After obtaining the self-attention result, the Add & Norm operation is performed, and the result is fed to the feed forward network (FFN). Then, the Add & Norm operation is performed again, and the output of a layer of DAG-Transformer Encoder is obtained. Calculations of the FFN in DAG-Transformer Encoder can be expressed as the equation below.

$$FFN(X) = W_2 \mathrm{ReLU}(W_1 X + b_1) + b_2,$$

where W_1, W_2, b_1, b_2 are the weights and biases of the two fully connected layers in FFN. Then, the calculations in one DAG-Transformer Encoder layer can be expressed as:

$$\begin{aligned} attn &= \mathrm{self} - attention(X) \\ LN_1 &= \mathrm{LayerNorm}(X + attn) \\ LN_2 &= \mathrm{LayerNorm}(FFN(LN_1) + LN_1) \end{aligned}$$

Overall, in this phase (DAG-Transformer Encoder), the calculations can be given as:

$$\begin{aligned} Out &\in \mathrm{R}^{N_{task} \times d_k} \\ &= DAG - TransformerEncoder^{(\ell)}(X), \end{aligned}$$

where ℓ denotes the number of layers of the DAG-Transformer Encoder.

3.4.3.3. Classifier. After obtaining the output of the last layer of the DAG-Transformer Encoder, we perform a Global Average Pooling operation on its node dimension, which can be expressed as:

$$\begin{aligned} &\mathrm{GAP}(Out) \in \mathrm{R}^{d_k} \\ &= \frac{1}{N_{task}} \sum_{i=1}^{N_{task}} Out_{d_k}(i). \end{aligned}$$

Then, the result of Global Average Pooling operation is sent to a fully connected layer as a classifier, and the output logits are:

$$\begin{aligned} logits &\in \mathrm{R}^3 \\ &= W\mathrm{GAP}(Out) + b, \end{aligned}$$

where W and b are the weight and bias of the fully connected layer, respectively. Finally, logits are converted to classification probability through the softmax activation function, which can be expressed as:

$$y = \mathrm{softmax}(logits),$$

where y_i denotes the probability that the model predicts the performance of this workflow's unknown task (i.e., prediction target) into the i-th level.

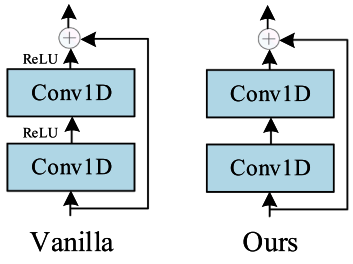


Fig. 12. A comparison between vanilla ResNet (left) and ours (right).

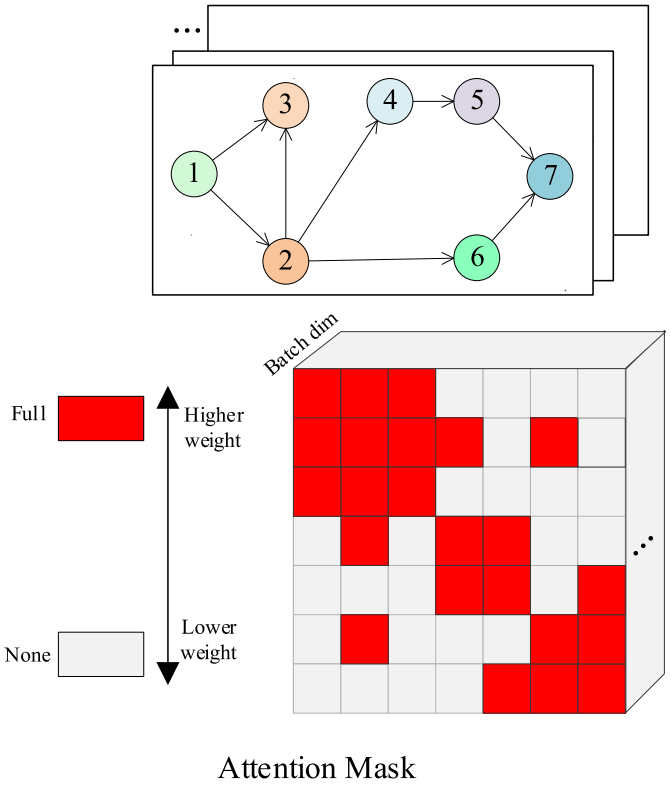


Fig. 13. How the attention mask works. In workflow operations, the information of a task node's use of resources has the greatest relationship with its neighboring nodes. An attention mask allows a single node to only calculate the attention of itself and its neighboring nodes when the model performs attention calculations, and masks other nodes with less impact.

4. Experiments

We conduct systematic experiments on the workflows with 7 tasks due to their sufficient samples and diversity of structures.

4.1. Training strategy and experimental environment

The training strategies used in experiments are listed in Table 6.

The Adam optimizer has a series of advantages. It can automatically adjust the learning rate of the parameters, and significantly improve training speed and stability. We choose the cross entropy loss function to match the softmax output, which is standard for classification problems.

The widely used stepwise or cosine annealing learning rate control method has a large initial learning rate, which often leads to training instability or collapse during the initial training process. But a small initial learning rate causes the model to enter an early local optimal state, thereby degrading the final performance. To this end, we design a TriangleStepWise learning rate control method based on the OCL learning rate control method [69], which gradually increases the learning rate from near zero linearly in the early training stage to the maximum learning rate (a.k.a., warmup), and decreases the learning rate linearly in rest training stage.

The TriangleStepWise learning rate control method is shown in Fig. 14 and can be expressed as:

$$\begin{cases} x \times \Delta lr & x < e \\ e \times \Delta lr - (x - e) \times \Delta lr & x \geq e \end{cases},$$

where $\Delta lr = max\_lr / \left(\frac{total\_epochs}{2} \right)$, $e = total\_epochs/2$.

The hardware and software configuration of computational environment used in all the experiments is listed in Table 7:

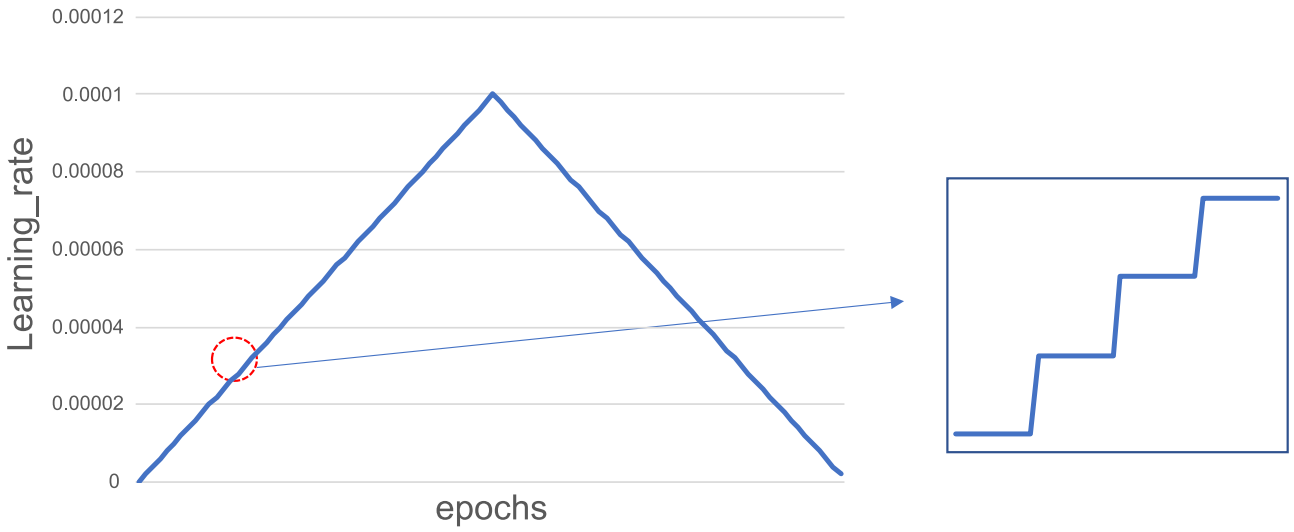


Fig. 14. Learning rate control method

Table 6
"training strategies" table.

Training strategy	specific method
optimizer	Adam
Loss	CrossEntropy
lr_scheduler	TriangleStepWise

Table 7
Computational Environment

programming language/environment	version/type
GPU	RTX 3080
CPU	Intel core i7-10700K
RAM	64GiB
CUDA	11.0
Python	3.8
Pytorch	1.7.0

We choose the top-1 accuracy as the evaluation metric in the experiments. Top-1 accuracy can be calculated using the following equation.

$$\text{Accuracy} = \frac{\text{num}(label\_pred = label\_true)}{\text{num}(label)} \times 100\%,$$

where *label_pred* indicates the classification result inferenced by the model, *label_true* indicates the ground truth of the dataset, $num(label\_pred = label\_true)$ indicates the correct number of classifications, and $num(label)$ indicates the total number of workflows in the dataset.

4.2. Descriptions of the experiments

We design three sets of experiments.

In Experiment 1, according to the requirements of the cloud workflow scheduling algorithm, we explore the combination of the following conditions: workflows with different structures, tasks in different context of workflows, the amount of available historical tasks' performance, and whether DAG structure is used.

In Experiment 2, we test our model's convergence speed.

In Experiment 3, we test the overall performance of our proposed model.

We perform three different splits of the dataset in three proportions: split(9-0.5-0.5), split(8-1-1), and split(6-2-2). Split(6-2-2) is used in Experiment 1, while all splits are used in Experiment 2 and Experiment 3.

We also implement several mainstream DL methods (LSTM, CNN, vanilla Transformer) and graph convolutional network (GCN), and compare them with the DAG-Transformer in Experiment 2 and 3. Please note that due to the GCN's special semi-supervised learning style, we do not add it to the comparison in Experiment 2.

Before the data is fed into prediction models, we mask all unknown tasks' information to 0.

4.3. Results

4.3.1. Experiment 1

In Experiment 1, we predict the 3rd, 5th, and 7th task's CPU and memory performance using DAG-Transformer, and explore the influence of the amount of known historical information (all previous tasks' information or the latest task's information) and whether the DAG structure information is known on the prediction results. The results are shown in Table 8.

4.3.2. Experiment 2

In Experiment 2, we compare different DL models' convergence speeds. The training parameters are shown in Table 9. In each model, we adjust their hyper-parameters to maximize their performance. The results of Experiment 2 are listed in Table 10. To show the convergence speed of each model more intuitively, the accuracy rate curve and loss curve are shown in Fig. 15.

4.3.3. Experiment 3

In Experiment 3, we give the best accuracy of each model and the results are as shown in Table 11.

4.3.4. Computational Overheads.

The hidden layer dimension of LSTM is 1024, and the number of layers is 6. The hidden layer dimension of the DAG-Transformer is 1024, and the number of layers is 6. The hidden dims of GCN's 3 graph-convolutional layer are 64, 128, 256 respectively. According to the semi-supervised learning, we put the whole graph into the (GPU)memory to train the GCN, so there is no difference in inference time between different *batch_size*. It is worth noting that, batched GCN performs worse, so we omitted it. The results are shown in Table 12.

Table 8
"results of experiment1" table.

Predicted target	Predicted task	All previous tasks' information	The latest task's information	With Graph Structure or Not	Accuracy
CPU	3rd	√	×	Yes	86.91%
				No	81.24%
		×	√	Yes	84.80%
				No	77.34%
	5th	√	×	Yes	85.68%
				No	78.70%
		×	√	Yes	82.37%
				No	69.53%
	7th	√	×	Yes	91.22%
				No	88.92%
		×	√	Yes	88.71%
				No	84.99%
Memory	3rd	√	×	Yes	94.51%
				No	92.77%
		×	√	Yes	93.62%
				No	91.48%
	5th	√	×	Yes	96.68%
				No	93.17%
		×	√	Yes	95.93%
				No	92.32%
	7th	√	×	Yes	98.56%
				No	97.12%
		×	√	Yes	97.44%
				No	94.86%

Table 9
"parameters settings" table.

parameters settings	value
batch_size	500
max_lr	1e-4
total_epochs	100
Evaluation metric	Accuracy rate

4.4. Discussion of experiments

In Experiment 1, we observe that, overall, the prediction accuracies decrease with following combinations: DAG structure + all previous tasks' information > DAG structure + the latest task's information > No DAG structure + all previous tasks' information > No DAG structure + the latest task's information. The effective use of DAG structure information consistently improves the performance of the prediction model, verifying the importance of the DAG structure. Moreover, DAG structure information is more necessary when available historical performance information is limited. When DAG structure information is available, our model can still achieve more than 80% CPU prediction accuracy even if using only the latest task's performance information.

From the perspective of scheduling in the cloud workflow, the DAG structures of batch workflows are generally known before execution. Therefore, the DAG information can be combined into the prediction model naturally, which will greatly improve the prediction performance.

In Experiment 2 and Computational Overheads (in *4.3.4*), we see that DAG-Transformer is better than the traditional deep learning models in terms of the convergence speed and accuracy while maintaining a relatively low overhead. Specifically, we find that the training curve of vanilla Transformer fluctuates drastically. This shows that although Transformer and self-attention have been applied in many fields successfully, vanilla Transformer cannot migrate seamlessly to all domains. "Attention is indeed all you need", but Transformer need more customization to fit specific problems and data structures.

In Experiment 3, the superiority of our method is systematically verified by comparing it with traditional machine learning methods (e.g., Linear Regression, XGBoost), deep learning (e.g., CNN, LSTM), and graph neural networks (GCN).

We find that the existing methods are insufficient in complex feature interaction learning. The overall performance is ranked from high to low as DAG-Transformer > GCN > LSTM > CNN. The convolution operation in CNN is a specific information extraction method designed for image data, and is not suitable for data with graph structure like workflow; LSTM is a recurrent neural network for sequence learning, which is hard to characterize the parallelism and dependency between tasks in the workflow. We also add GCN for comparison. We find that using bi-directional edges in GCN can improve the model's perception ability of workflow data compared to using unidirectional edges, and thus achieve better prediction accuracy. The relatively good performance of GCN shows that if the graph structure information is well utilized, better accuracy can be achieved despite the simplicity of its network architecture.

DAG-Transformer, which can adaptively focus the nonlinear high-order features, takes advantage of the sequential and graph relationships of workflows to enhance the feature embedding. We infer that the DAG-Transformer could effectively learn the following implicit relationships in the workflow. 1) There should be some similarities in the performance of each task in the workflow with the same quantity of tasks and the same graph structure. 2) There should be some similarities in the task's performance at the same position (represented by the position encoding) in the workflow. 3) In the workflow, the predecessor tasks' performance usually has correlations with the successor tasks' performance.

5. Conclusions and Future Works

5.1. Conclusions

Under offline-online hybrid co-location deployment mode, new architectures such as containers and microservices, show a universal trend and great superiority in cloud computing applications like big data and artificial intelligence, but bring huge complexity in management and optimization. To address the significant resource supply-demand mismatch problem in cloud scheduling, this study transforms cloud computing logs into graph spatiotemporal sequences, proposes a workflow performance prediction model, and gives a novel end-to-end deep

Table 10
"results of experiment2" table.

Model	Accuracy in split(9-0.5-0.5)	Significant test compared with DAG-Transformer		Accuracy in split (8-1-1)	Significant test compared with DAG-Transformer		Accuracy in split (6-2-2)	Significant test compared with DAG-Transformer	
		t-statistic	P(T<=t) single tail		t-statistic	P(T<=t) single tail		t-statistic	P(T<=t) single tail
DAG-Transformer	**90.40%±0.25%**	-	-	**89.20%±0.08%**	-	-	**89.67%±0.07%**	61.8	2.05e-07
CNN	82.57%±0.05%	60.0	2.3e-07	81.44%±0.03%	168.2	7.04e-11	81.63%±0.03%	44.5	4.31e-09
LSTM	87.19%±0.10%	23.3	1.35e-06	86.21%±0.07%	53.0	8.85e-12	85.84%±0.15%	37.3	1.54e-06
Vanilla Transformer	81.46%±0.40%	37.2	1.32e-09	80.50%±0.45%	37.8	1.46e-06	80.36%±0.49%	61.8	72.05e-07

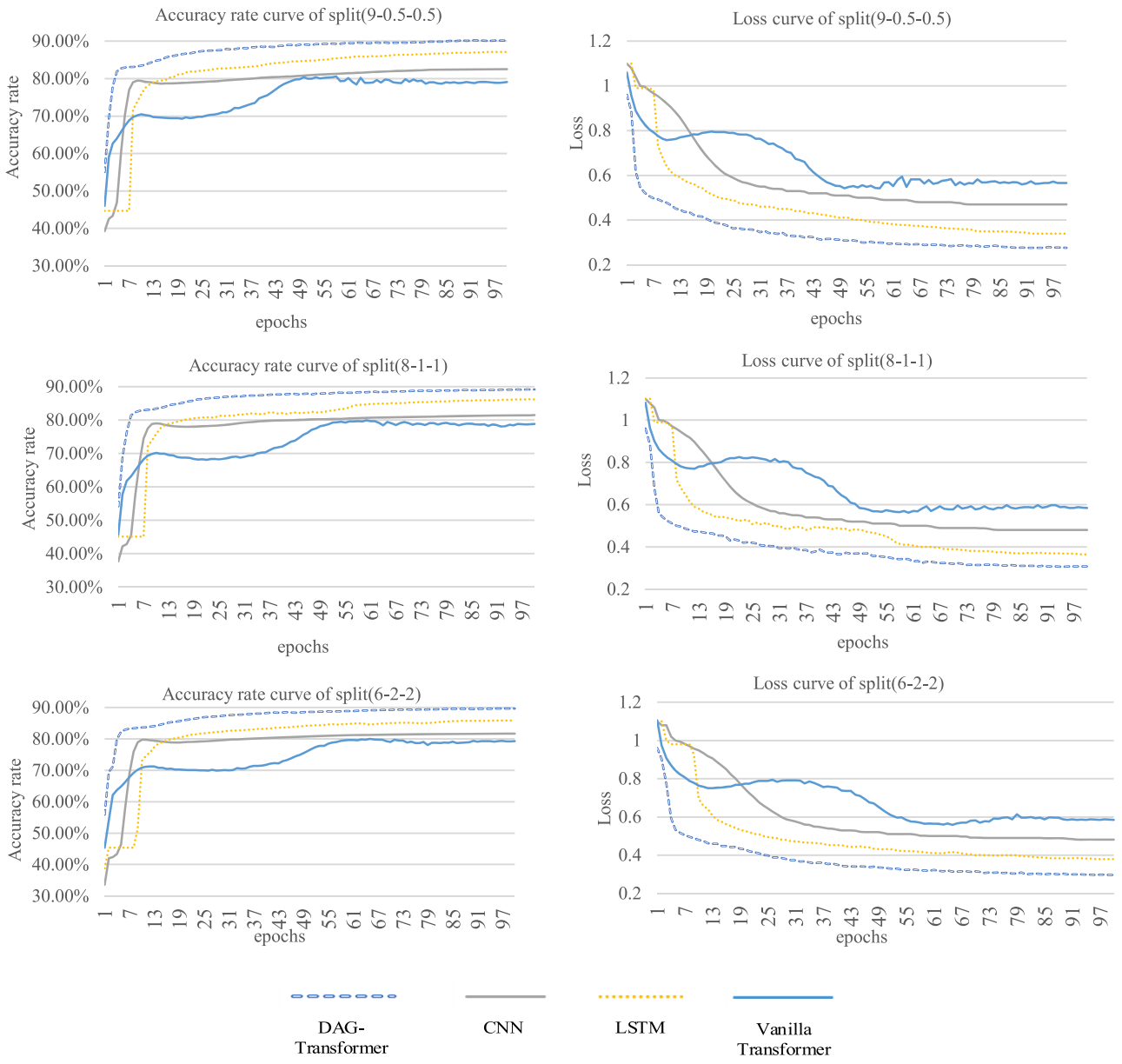


Fig. 15. Accuracy rate curve and loss curve in experiment 2

Table 11
"results of experiment3" table (The parameters and architecture of all models were adjusted to the best.)

Model	Accuracy in split(9-0.5-0.5)	Significant test compared with DAG-Transformer		Accuracy in split (8-1-1)	Significant test compared with DAG-Transformer		Accuracy in split (6-2-2)	Significant test compared with DAG-Transformer	
		t-statistic	P(T<=t) single tail		t-statistic	P(T<=t) single tail		t-statistic	P(T<=t) single tail
DAG-Transformer	**92.15%±0.13%**	-	-	**91.11%±0.05%**	-	-	**91.25%±0.04%**	-	-
CNN	86.98%±0.10%	62.6	3.48e-11	86.56%±0.03%	138.4	1.36e-13	86.62%±0.06%	119.3	1.36e-14
LSTM	88.92%±0.17%	29.8	6.23e-09	88.35%±0.20%	26.0	7.82e-07	88.44%±0.14%	38.4	1.12e-07
GCN(Unidirectional)	86.86%±0.48%	21.2	2.16e-06	86.06%±0.61%	16.4	4.05e-05	86.56%±0.55%	16.8	3.66e-05
GCN(Bi-directional)	89.40%±0.21%	22.0	5.12e-08	88.42%±0.18%	27.7	5.70e-07	88.78%±0.11%	39.5	9.79e-08
Vanilla Transformer	81.46%±0.40%	50.0	3.02e-08	80.50%±0.45%	46.7	6.30e-07	80.36%±0.49%	43.8	8.11e-07
SVM Classifier	83.58%±0.01%	129.7	1.06e-08	82.89%±0.00%	295.5	3.93e-10	83.46%±0.00%	316.9	2.97e-10
Xgboost	90.61%±0.00%	23.4	9.81e-06	90.02%±0.00%	39.3	1.26e-06	90.70%±0.00%	26.0	6.54e-06
Logistic Regression(l1/lasso)	83.52%±0.00%	131.1	1.01e-08	82.74%±0.00%	300.9	3.66e-10	83.30%±0.00%	323.4	2.74e-10
Logistic Regression(l2/ridge)	82.45%±0.00%	147.4	6.36e-09	81.74%±0.01%	331.7	2.48e-10	82.65%±0.00%	349.8	2.00e-10

Table 12
The computational overheads of training and inference (using (6-2-2)split)

Model	GPU RAM occupation	Training Time (in Experiment 3)	Inference Time (*batch_size*=500, device=GPU)	Inference Time (*batch_size*=1, device=GPU)	Inference Time (*batch_size*=500, device=CPU)	Inference Time (*batch_size*=1, device=CPU)
CNN	1565MiB	~25min 20s	0.00817ms/sample	0.757ms/sample	0.008ms/sample	0.624ms/sample
LSTM	4051MiB	~47min 17s	0.03415ms/sample	2.450ms/sample	1.069ms/sample	15.578ms/sample
GCN	7143MiB	~49min 43s	0.00297ms/sample		0.562ms/sample	
DAG-Transformer	2591MiB	~35min 26s	0.02969ms/sample	6.176ms/sample	0.356ms/sample	5.538ms/sample

Algorithm 1
Generation of Position Encoding

procedure Generation of Position Encoding
Inputs:
- Weighted out-degree adjacency matrix of a workflow A_1
- Number of tasks in a workflow N_{task}
- Dim of a node's attribute vector N_{feat}

generate a vector $p \in \mathrm{R}^{N_{task}}$ of all 0s.
signal←*True*
while *signal* $==$ *True*
temp←*p*
for m **in** $0,\ldots,N_{task}-1$ **do**
for n **in** $0,\ldots,N_{task}-1$ **do**
if $A_1[m][n]! = 0$ **do**
$p[n] \leftarrow \max(p[n], p[m]+1)$
end if
end for
end for
if *temp* $==$ p **do**
signal←*False*
end if
end while
generate a matrix $P \in \mathrm{R}^{N_{task} \times N_{feat}}$
$P_{x,2h} = \sin(p[x]/10000^{2h/N_{feat}}) P_{x,2h+1} = \cos(p[x]/10000^{2h/N_{feat}})$
Output: -Position Encoding Matrix P
end procedure

Algorithm 2
Generation of Attention Mask

procedure Generation of Attention Mask
Inputs:
- Normalized out-degree adjacency matrix of a workflow A_1
- In-degree adjacency matrix of a workflow A_2
- Number of tasks in a workflow N_{task}

$M = A_1 + A_2 + I$(I is the identity matrix)
for i **in** $0,\ldots,N_{task}-1$ **do**
for j **in** $0,\ldots,N_{task}-1$ **do**
if $M[i][j]! = 0$ **do**
$M[i][j] \leftarrow -\inf$
else do
$M[i][j] \leftarrow 0$
end if
end for
end for
Output:- Attention Mask M
end procedure

learning neural network. Our research provides a new way to facilitate workflow scheduling by data-driven predictions.

Our main contributions include the following:

1 We systematically investigate the research status of cloud computing, present challenges for workflows scheduling in new generation cloud environment, and the new opportunity for data-driven prediction approach. Combining the existing qualitative analysis from literature and our exploratory analysis on Cluster-trace-v2018, we aggregate tasks as well as their DAG information into graph-based workflow dataset, and give the definition of labels based on clustering. Finally, a graph-based sequence prediction model is established, which provides a paradigm to model workflow-oriented predictions problem under the scenario of containerized resource management. Our proposed approach is also suitable for general- purpose prediction problems of industrial process. We also publish the dataset used in this paper.

2 We propose DAG-Transformer which seamlessly integrate workflow DAG structure information to learn from the graph spatiotemporal sequence data. DAG-Transformer is able to auto-focus on nonlinear high-order features interaction through attention mechanism, and takes advantage of the sequential and graph relationships of workflows to enhance the feature embedding. We give the detailed description of the core components (attention mask, position encoding) and the overall neural network architecture. The superiority of our method is verified by systematic experiments and comparisons with classic machine learning methods (e.g., Linear Regression, XGBoost), deep learning (e.g., CNN, LSTM), and graph neural networks (GCN). We find that the existing ML methods are insufficient in complex graph learning, and emphasize the necessity of our method.

3 According to the requirements of the cloud workflow scheduling algorithm, we conduct systematic experiments on the combination of the following conditions: workflows with different structures, tasks in different contexts of workflows, the amount of available historical tasks' performance, and whether the DAG structure is used. We find that the effective use of DAG structure information consistently improves the performance of the prediction model, which verifies the importance of the DAG structure. Even in the initial execution phase of the workflow, where available historical performance information is limited, our model still achieves more than 80% CPU prediction accuracy and 90% memory prediction accuracy. Our model and exploration, for the first time, highlights the potential of prediction for scheduling in cloud workflows from a quantitative perspective, giving a clear and intuitive baseline.

5.2. Future Works

With the arrival of the Internet of Everything in the 5G era [70], large and complex networks of data relationships are woven into the production and operation processes of smart manufacturing, logistics and supply chain, transportation, and Internet of Things industries. For instance, in the context of Industry 4.0, process automation and predictive maintenance play an essential role [71]. Hence, predictions under such circumstances contribute to both operations and maintenances. The application prospect of our method is fit to not only cloud computing scenarios, but also similar industrial scenarios with graph structure information. For example, future indicators predictions based on historical information or unknown indicators imputation in the absence of sensors.

In future work, our research will investigate self-supervised or semi-supervised pre-training to apply transfer learning across heterogeneous workflows with different number of nodes, structures, and indicators to improve the overall accuracy and robustness of prediction. We will also try to explore methods for multi-node simultaneous prediction.

Data and Code Availability

The data and model used in this study is available at https://github.com/cloudworkflow/workflow-performance-prediction-jii. For more details, please contact the corresponding author.

Credit author statement

All authors contributed equally.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (71772033, 71831003, 71801031, 72172025), Natural Science Foundation of Liaoning Province, China (Joint Funds for Key Scientific Innovation Bases, 2020-KF-11-11), and also supported by a grant from the Department of Industrial and Systems Engineering of the Hong Kong Polytechnic University (H-ZG3K).

Appendices

Figure of the number of workflow structures and the number of tasks:

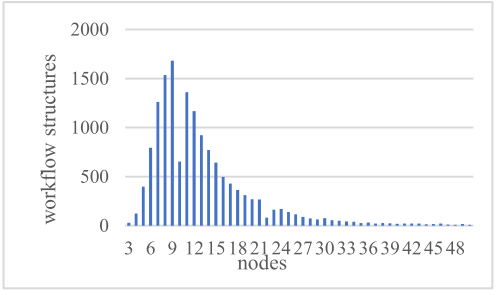


Figure of the number of tasks in workflow and the number of tasks:

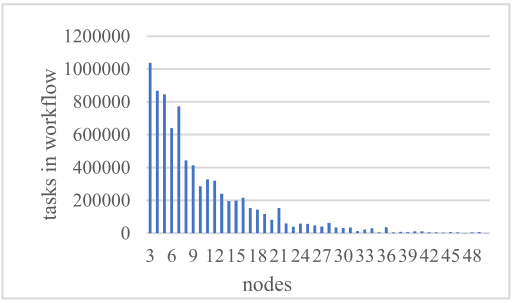


Figure of the number of workflow samples and the number of tasks:

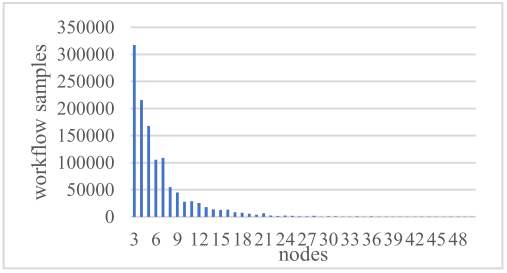
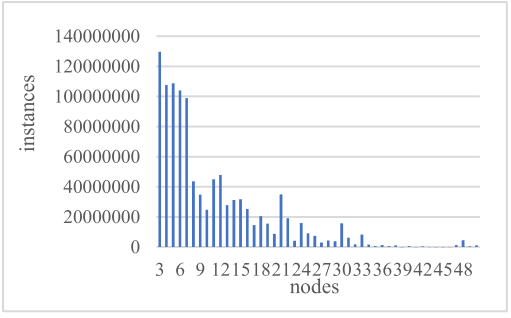


Figure of the number of instances and the number of tasks:



"workflow-task-features" table.

	Feature Name	Interpretation	Feature Name	Interpretation
Other	count	Number of instances included in the task		
Resource occupation	mean_ca	Average of CPU average usage	mean_cm	Average of CPU maximum usage
	var_ca	Variance of CPU average usage	var_cm	Variance of CPU maximum usage
	max_ca	Maximum of CPU average usage	max_cm	Maximum of CPU maximum usage
	min_ca	Minimum of CPU average usage	min_cm	Minimum of CPU maximum usage
	med_ca	Median of CPU average usage	med_cm	Median of CPU maximum usage
	skew_ca	Skewness of CPU average usage	skew_cm	Skewness of CPU maximum usage
	kurt_ca	Kurtosis of CPU average usage	kurt_cm	Kurtosis of CPU maximum usage
	mean_ma	Average of memory average usage	mean_mm	Average of memory maximum usage
	var_ma	Variance of memory average usage	var_mm	Variance of memory maximum usage
	max_ma	Maximum of memory average usage	max_mm	Maximum of memory maximum usage
	min_ma	Minimum of memory average usage	min_mm	Minimum of memory maximum usage
	med_ma	Median of memory average usage	med_mm	Median of memory maximum usage
	skew_ma	Skewness of memory average usage	skew_mm	Skewness of memory maximum usage
	kurt_ma	Kurtosis of memory average usage	kurt_mm	Kurtosis of memory maximum usage
Lifecycle	mean_t	Average running time	max_t	Maximum running time
	var_t	Variance of running time	min_t	Minimum running time
	maxtime	Actual running time		

"Task's DAG features in workflow" table.

Feature Name	Interpretation	Feature Name	Interpretation
o_1	Out degree to task 1	i_1	In degree to task 1
o_2	Out degree to task 2	i_2	In degree to task 2
o_3	Out degree to task 3	i_3	In degree to task 3
o_4	Out degree to task 4	i_4	In degree to task 4
o_5	Out degree to task 5	i_5	In degree to task 5
o_6	Out degree to task 6	i_6	In degree to task 6
o_7	Out degree to task 7	i_7	In degree to task 7

References

[1] J. Shah, D. Dubaria, Building modern clouds: using docker, kubernetes & Google cloud platform. 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2019, pp. 0184–0189.

[2] U. Deshpande, D. Chan, S. Chan, K. Gopalan, N. Bila, Scatter-gather live migration of virtual machines, IEEE Transactions on Cloud Computing 6 (1) (2015) 196–208.

[3] A.M. Manasrah, H. Ba Ali, Workflow scheduling using hybrid GA-PSO algorithm in cloud computing, Wireless Communications and Mobile Computing 2018 (2018).

[4] L.I.N. Bing, G.U.O. Wenzhong, C.H.E.N. Guolong, Scheduling strategy for science workflow with deadline constraint on multi-cloud, Journal on Communications 39 (1) (2018) 56.

[5] M. Gao, M. Chen, A. Liu, W.H. Ip, K.L. Yung, Optimization of microservice composition based on artificial immune algorithm considering fuzziness and user preference, IEEE Access 8 (2020) 26385–26404.

[6] C. Jiang, Y. Qiu, W. Shi, Z. Ge, J. Wang, S. Chen, J. Lin, Characterizing Co-located Workloads in Alibaba Cloud Datacenters, IEEE Transactions on Cloud Computing (2020).

[7] Q. Yang, Y. Zhou, Y. Yu, J. Yuan, X. Xing, S. Du, Multi-step-ahead host load prediction using autoencoder and echo state networks in cloud computing, The Journal of Supercomputing 71 (8) (2015) 3037–3053.

[8] R. Shaw, E. Howley, E. Barrett, Predicting the available bandwidth on intra cloud network links for deadline constrained workflow scheduling in public clouds. International Conference on Service-Oriented Computing, Springer, Cham, 2017, pp. 221–228.

[9] W. Zhong, Y. Zhuang, J. Sun, J. Gu, A load prediction model for cloud computing using PSO-based weighted wavelet support vector machine, Applied Intelligence 48 (11) (2018) 4072–4083.

[10] A. Aslam, H. Chen, J. Xiao, H. Jin, Reasoning Based Workload Performance Prediction in Cloud Data Centers. 2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2019, pp. 431–438.

[11] Y. Zhu, W. Zhang, Y. Chen, H. Gao, A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment, EURASIP Journal on Wireless Communications and Networking 2019 (1) (2019) 1–18.

[12] W. Wei, X. Fan, H. Song, X. Fan, J. Yang, Imperfect information dynamic stackelberg game based resource allocation using hidden Markov for cloud computing, IEEE Transactions on Services Computing 11 (1) (2016) 78–89.

[13] K. Mason, M. Duggan, E. Barrett, J. Duggan, E. Howley, Predicting host CPU utilization in the cloud using evolutionary neural networks, Future Generation Computer Systems 86 (2018) 162–173.

[14] Aldossary, M., & Djemame, K. (2018, March). Performance and Energy-based Cost Prediction of Virtual Machines Live Migration in Clouds. In CLOSER (pp. 384-391).

[15] Y. Yu, V. Jindal, F. Bastani, F. Li, I.L. Yen, Improving the smartness of cloud management via machine learning based workload prediction. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), IEEE, 2018, pp. 38–44. Vol. 2.

[16] W. Tan, L. Huang, M.Y. Kataev, Y. Sun, L. Zhao, H. Zhu, N. Xie, Method towards reconstructing collaborative business processes with cloud services using evolutionary deep Q-learning, Journal of Industrial Information Integration 21 (2021), 100189.

[17] S. Di, D. Kondo, W. Cirne, Characterization and comparison of cloud versus grid workloads. 2012 IEEE International Conference on Cluster Computing, IEEE, 2012, pp. 230–238.

[18] L.D. Dhinesh Babu, A. Gunasekaran, P.V Krishna, A decision-based pre-emptive fair scheduling strategy to process cloud computing work-flows for sustainable enterprise management, International Journal of Business Information Systems 16 (4) (2014) 409–430.

[19] M.A. Rodriguez, R. Buyya, Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds, IEEE transactions on cloud computing 2 (2) (2014) 222–235.

[20] S. AlEbrahim, I. Ahmad, Task scheduling for heterogeneous computing systems, The Journal of Supercomputing 73 (6) (2017) 2313–2338.

[21] A. Atef, T. Hagras, Y.B. Mahdy, J. Janeček, Lower-bound complexity algorithm for task scheduling on heterogeneous grid, Computing 99 (11) (2017) 1125–1145.

[22] J. Du, L. Zhao, J. Feng, X. Chu, Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee, IEEE Transactions on Communications 66 (4) (2017) 1594–1608.

[23] M. Duggan, R. Shaw, J. Duggan, E. Howley, E. Barrett, A multitime-steps-ahead prediction approach for scheduling live migration in cloud data centers, Software: Practice and Experience 49 (4) (2019) 617–639.

[24] 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.

[25] J. Prassanna, N. Venkataraman, Adaptive regressive holt–winters workload prediction and firefly optimized lottery scheduling for load balancing in cloud, Wireless Networks 27 (8) (2021) 5597–5615.

[26] N. Suksriupatham, A. Hoonlor, Workload Prediction with Regression for Over and Under Provisioning Problems in Multi-agent Dynamic Resource Provisioning Framework. 2020 17th International Joint Conference on Computer Science and Software Engineering (JCSSE), IEEE, 2020, pp. 128–133.

[27] S.Y. Hsieh, C.S. Liu, R. Buyya, A.Y. Zomaya, Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers, Journal of Parallel and Distributed Computing 139 (2020) 99–109.

[28] Z. Xiao, B. Wang, X. Li, J. Du, Workload-driven coordination between virtual machine allocation and task scheduling, Neural Computing and Applications 32 (10) (2020) 5535–5551.

[29] Marahatta, A., Xin, Q., Chi, C., Zhang, F., & Liu, Z. (2020). PEFS: AI-driven Prediction based Energy-aware Fault-tolerant Scheduling Scheme for Cloud Data Center. IEEE Transactions on Sustainable Computing.

[30] C. Li, J. Bai, Y. Luo, Efficient resource scaling based on load fluctuation in edge-cloud computing environment, The Journal of Supercomputing (2020) 1–32.

[31] F. Davami, S. Adabi, A. Rezaee, A.M. Rahmani, Distributed scheduling method for multiple workflows with parallelism prediction and DAG prioritizing for time constrained cloud applications, Computer Networks 201 (2021), 108560.

[32] G. Kaur, A. Bala, Prediction based task scheduling approach for floodplain application in cloud environment, Computing 103 (5) (2021) 895–916.

[33] Yeung, G., Borowiec, D., Yang, R., Friday, A., Harper, R., & Garraghan, P. (2021). Horus: Interference-Aware and Prediction-Based Scheduling in Deep Learning Systems. IEEE Transactions on Parallel and Distributed Systems.

[34] Y. Jiang, C.S. Perng, T. Li, R.N. Chang, Cloud analytics for capacity planning and instant VM provisioning, IEEE Transactions on Network and Service Management 10 (3) (2013) 312–325.

[35] D. Janardhanan, E. Barrett, CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models. 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, 2017, pp. 55–60.

[36] S. Gupta, A.D. Dileep, T.A. Gonsalves, A joint feature selection framework for multivariate resource usage prediction in cloud servers using stability and prediction performance, The Journal of Supercomputing 74 (11) (2018) 6033–6068.

[37] Q. Zhang, L.T. Yang, Z. Yan, Z. Chen, P. Li, An efficient deep learning model to predict cloud workload for industry informatics, IEEE transactions on industrial informatics 14 (7) (2018) 3170–3178.

[38] Erradi, A., Iqbal, W., Mahmood, A., & Bouguettaya, A. (2019). Web application resource requirements estimation based on the workload latent features. IEEE Transactions on Services Computing.

[39] Fei, B., Zhu, X., Liu, D., Chen, J., Bao, W., & Liu, L. (2020). Elastic resource provisioning using data clustering in cloud service platform. IEEE Transactions on Services Computing.

[40] J. Gao, H. Wang, H. Shen, Machine learning based workload prediction in cloud computing. 2020 29th international conference on computer communications and networks (ICCCN), IEEE, 2020, pp. 1–9.

[41] G. Rjoub, J. Bentahar, O. Abdel Wahab, A. Saleh Bataineh, Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems, Concurrency and Computation: Practice and Experience (2020) e5919.

[42] H.A. Kholidy, An intelligent swarm based prediction approach for predicting cloud computing user resource needs, Computer Communications 151 (2020) 133–144.

[43] J. Bi, S. Li, H. Yuan, M. Zhou, Integrated deep learning method for workload and resource prediction in cloud systems, Neurocomputing 424 (2021) 35–48.

[44] M.E. Karim, M.M.S. Maswood, S. Das, A.G. Alharbi, BHyPreC: A Novel Bi-LSTM based Hybrid Recurrent Neural Network Model to Predict the CPU Workload of Cloud Virtual Machine, IEEE Access 9 (2021) 131476–131495.

[45] F. Wu, Q. Wu, Y. Tan, Workflow scheduling in cloud: a survey, The Journal of Supercomputing 71 (9) (2015) 3373–3418.

[46] G. Kousalya, P. Balakrishnan, C.P. Raj, Workflow Predictions Through Operational Analytics and Machine Learning. Automated Workflow Scheduling in Self-Adaptive Clouds, Springer, Cham, 2017, pp. 119–135.
[47] M. Gao, Y. Li, J. Yu. Workload Prediction of Cloud Workflow Based on Graph Neural Network. In International Conference on Web Information Systems and Applications, Springer, Cham, 2021, September, pp. 169–189.
[48] Github, Google cluster trace. https://github.com/google/cluster-data, 2019 (accessed 15 July 2020).
[49] Github, Alibaba cluster trace program. https://github.com/alibaba/clusterdata, 2018 (accessed 15 July 2020).
[50] Liu, Q., & Yu, Z. (2018, October). The elasticity and plasticity in semi-containerized co-locating cloud workload: a view from alibaba trace. In Proceedings of the ACM Symposium on Cloud Computing (pp. 347-360).
[51] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, Y. Bao, Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces. 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), IEEE, 2019, pp. 1–10.
[52] L. Deng, Y.L. Ren, F. Xu, H. He, C. Li, Resource utilization analysis of Alibaba cloud. International conference on Intelligent Computing, Springer, Cham, 2018, pp. 183–194.
[53] Cheng, Y., Chai, Z., & Anwar, A. (2018, August). Characterizing co-located datacenter workloads: An alibaba case study. In Proceedings of the 9th Asia-Pacific Workshop on Systems (pp. 1-3).
[54] Lu, C., Chen, W., Ye, K., & Xu, C. Z. (2020, May). Understanding the Workload Characteristics in Alibaba: A View from Directed Acyclic Graph Analysis. In 2020 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS) (pp. 1-8). IEEE.
[55] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems 25 (2012) 1097–1105.
[56] A. Graves, A.R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks. 2013 IEEE international conference on acoustics, speech and signal processing, IEEE, 2013, pp. 6645–6649.
[57] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.
[58] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv: 2005.14165.
[59] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, D. Hassabis, Highly accurate protein structure prediction with AlphaFold, Nature 596 (7873) (2021) 583–589.
[60] S. Ravuri, K. Lenc, M. Willson, D. Kangin, R. Lam, P. Mirowski, S. Mohamed, Skillful Precipitation Nowcasting using Deep Generative Models of Radar, Nature 597 (2021) 672–677.
[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).
[62] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv: 1810.04805.
[63] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.
[64] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030.
[65] J.A. Hartigan, M.A. Wong, Algorithm AS 136: A k-means clustering algorithm, Journal of the royal statistical society. series c (applied statistics) 28 (1) (1979) 100–108.
[66] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, Journal of machine learning research 9 (11) (2008).
[67] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, E. Duchesnay, Scikit-learn: Machine learning in Python. the, Journal of machine Learning research 12 (2011) 2825–2830.
[68] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
[69] L.N. Smith, N. Topin, Super-convergence: Very fast training of neural networks using large learning rates. Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, International Society for Optics and Photonics, 2019, 1100612. Vol. 11006.
[70] S. Li, L. Da Xu, S. Zhao, 5G Internet of Things: A survey, Journal of Industrial Information Integration 10 (2018) 1–9.
[71] E.G. Gorski, E.D.F.R. Loures, E.A.P. Santos, R.E. Kondo, G.R.D.N. Martins, Towards a smart workflow in CMMS/EAM systems: An approach based on ML and MCDM, Journal of Industrial Information Integration (2021), 100278.
[72] Tian, H., Zheng, Y., & Wang, W. (2019, November). Characterizing and synthesizing task dependencies of data-parallel jobs in alibaba cloud. In Proceedings of the ACM Symposium on Cloud Computing (pp. 139-151).