Scientific Workflow Makespan Minimization In Edge Multiple Service Providers Environment

S. Sabahat H. Bukhari
Neijiang Normal University
Muhammad Usman Younus (usman1644@gmail.com)
Universite Federale Toulouse Midi-Pyrenees Ecole Doctorale Mathematiques Informatique Telecommunications de Toulouse https://orcid.org/0000-0001-9033-1767
Zain-ul-Abidin Jaffari
NNU: Northwest Nazarene University
Muhammad Arshad Shehzad Hassan
The University of Faisalabad
Muhammad Rizwan Anjum
The Islamia University of Bahawalpur
Sanam Narejo
Mehran University of Engineering and Technology

Research Article

Keywords: Edge Computing, Workflow, MakeSpan, Multiple service Provider

Posted Date: June 2nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-465640/v1

Scientific Workflow Makespan Minimization in Edge Multiple Service Providers Environment

S. Sabahat H. Bukhari[1], Muhammad Usman Younus* [2], Zain-ul-Abidin Jaffari[3], Muhammad Arshad Shehzad Hassan[4], Muhammad Rizwan Anjum[5], Sanam Narejo [6]

1 College of Computer Science, Neijiang Normal University, Neijiang, 641100, P.R. China.
2 Ecole Mathématiques, Informatique, Télécommunications de Toulouse, Université de Toulouse, Toulouse, France.
3 College of Physics and Electronic Information Engineering, Neijiang Normal University, Neijiang, 641100, P.R. China
4 Department of Engineering Technology, The University of Faisalabad, Faisalabad 38000, Pakistan
5 Department of Electronic Engineering, The Islamia University of Bahawalpur, Bahawalpur ,63100, Pakistan.
6 Department of Computer Systems Engineering, Mehran University of Engineering & Technology (MUET), Jamshoro, Pakistan
*Corresponding Author: Muhammad Usman Younus. Email: usman1644@gmail.com
Muhammad Rizwan Anjum . Email: engr.muhammadrizwan@gmail.com
S. Sabahat H. Bukhari. Email: sabahatbukhari@njtc.edu.cn

Abstract

The edge computing model offers an ultimate platform to support scientific and real-time workflow-based applications over the edge of the network. However, scientific workflow scheduling and execution still facing challenges such as response time management and latency time. This leads to deal with the acquisition delay of servers, deployed at the edge of a network and reduces the overall completion time of workflow. Previous studies show that existing scheduling methods consider the static performance of the server and ignore the impact of resource acquisition delay when scheduling workflow tasks. Our proposed method presented a meta-heuristic algorithm to schedule the scientific workflow and minimize the overall completion time by properly managing the acquisition and transmission delays. We carry out extensive experiments and evaluations based on commercial clouds and various scientific workflow templates. The proposed method has approximately 7.7% better performance than the baseline algorithms, particularly in overall deadline constraint that gives a success rate.

Keywords: Edge Computing; Workflow; MakeSpan; Multiple service Provider;

1. Introduction

Service centric paradigm of QoS and networking-based delivery provides plethora of new services. From a broader point of view, most of the hardware and software abilities have turn into consumable and deliverable services. In the past few years, cloud computing technologies such as computing and storage have been the ideal trends [1-4]. However, recently the technology has moved one step forward, and a new concept has emerged known as edge computing. In an edge environment, computing and storage devices are moves at the edge of the network, and the client can also access them via 3G, 4G, 5G, or Wi-Fi network. Because of this, transmission time and resource acquisition time between computational

devices and the end-user has been significantly reduced. Edge computational servers are very helpful for real-time and for scientific applications. In this scenario, the edge computational servers are the best option to schedule the scientific application's tasks. In scientific applications, the workflow scheduling acknowledged as an NP-hard problem [5] is one of the core issues because workflow tasks are interdependent and cannot be executed in a simple way [6]. Single scientific workflow usually contains numerous interdependent tasks that are tricky to schedule and difficult to perform at less time [7]. Edge computing environments typically own a number of computational machines (servers) provided by different service providers with dissimilar configurations, transmission, and acquisition time overhead [8, 9]. Many service providers offer heterogeneous servers and are exposed to edge users. In our scenario, any single instance offered by the service provider can fulfill the requirements of the workflow task, but the algorithm tends to select the server with minimum execution, acquisition, and transmission delay for the purpose of reducing the overall execution time of the workflow.

In the proposed methodology, we introduce a novel technique to manage the servers' transmission delay and performance variation. It employs a Genetic Algorithm for yielding near-optimal solutions with a constraint of workflow execution deadline. Our approach assumes that the servers on any access point are physically heterogeneous and different in performance wise; however, any server can execute the workflow tasks.

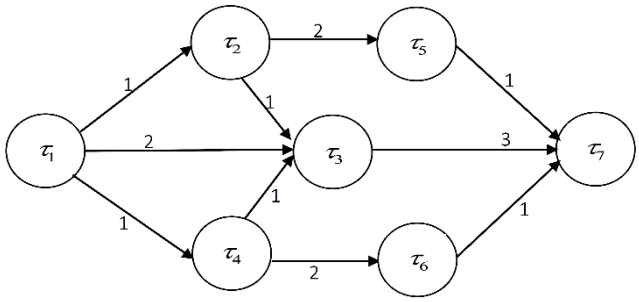

Figure 1 Example of workflow

In this article, our motivation is to present a metaheuristic algorithm to schedule the scientific workflow to minimize the overall completion time by appropriately managing the transmission and acquisition delays. Our proposed technique carefully chooses the resources offered by multiple service providers in the edge computing environment. We conduct an extensive evaluation based on real-world commercial clouds and scientific workflow templates (Huawei and Tencent). Our experimental results clearly show that our proposed method outperforms the baseline algorithms. Key Points of our article is mentioned below:

- Minimum scientific workflow makespan achieved based on deadline constraint in an edge environment
- All user's constraints are met in all scenarios
- Proposed Algorithms can perform better in multiple service providers Environment

The rest of the paper organized as follows. Section 2 illustrates the literature review of the work. Problem statement and methodology is illustrated in section 3. Section 4, is about a genetic algorithm for workflow makespan minimization in an edge computing environment (GA-WMM). Experiment and results analysis is discussed in section 5, and finally section 6 concludes the article.

2. Literature review

In the presented literature, technical features, pros and cons of edge computing have been extensively studied. The hybrid edge cloud can leverage to increase the performance of newly developed applications, e.g., healthcare and cloud gaming [10, 11]. Latency and power-aware cloudlets selection methodology is presented in [12] for task offloading in the multi-cloudlet environment. The proposed method presented in [13] tradeoff between service delay and power consumption in the fog-cloud system is examined, and workload allocation is performed by using the least energy cost under the latency constraint. Author in [14] considers the load balancing and optimizes the user to cloudlet assignment and cloudlet placement to reduce the service latency. Stackelnerg game and matching theory is presented in [15] to examine the optimization in three tire edge network. Three tire edge-networks include; (1) data service subscribers (DSS), (2) data service operators (DSO), and (3) a set of edge nodes. According to this study, DSOs can take computational resources from the different edge nodes to serve their DSSs.

The mobile edge computing environment is another research field that focuses on allocating the communication and computational resources for task offloading. Mobile edge computing permits the computational tasks to be offloaded to resource-rich servers located at cellular base stations. This can decrease the energy usage of the device and the task's execution time. In case the multiple users at the same time offload their number tasks to MEC computing instance, this can create a huge problem for the MEC. In this scenario, the user suffers extreme interference and can use a very small portion of the computing resources, which increases the transmission delay, high task execution time on the serves, and minimum data rate.

However, in edge computing, multiple users can offload their tasks to instances at the same time, and edge can offload tasks to the cloud through the core network in case of more computational power is required.

Real-time workflow scheduling problems based on mobile edge computing and mobile cloud computing schemes are presented in [16]. The author proposed a heterogeneous node network composed of three kinds of computing instances. A conventional data center is part of a heterogeneous node network, and cloudlets are situated in between the data center and edge server. However, the cloudlet is a computing instance with the lowest data transmission time between the edge server and cloudlet. However, the cloudlet has less computational power than the data center. It is useful for streaming workflow among, and it reduces the latency. In this study author also proposed a flexible heuristic-based streaming workflow scheduling to achieve the minimum cost. This proposed algorithm is a hybrid of bin-packing and shortest path algorithms in a graph network. Their results are compared with the linear programming, and it is proved that the solution is near-optimal and execution time is reduced.

A decentralized approach is presented in the article [17]. ISA95 architecture is used with RAMI 4.0 on edge computing and SOA-based systems. Workflow management is proposed to integrate MES data between the smart devices and smart products. In this model industrial use cases are demonstrated, such as (i) already planned model is used for scheduling and planning. (ii) Workflow manager is utilized for sequencing and coordination.

A workflow-net-based method is presented in [18] for mobile edge computing nodes which assure that the cloud network provides the media contents to SSOs. This approach offers a backtracking scheme that determines the minimum cost supportive path for the workflow-net. The author proved through demonstration that the system is scalable to a different number of nodes, and services can be provided with minimum cost.

The task scheduling algorithm is presented in [19], which is based on QoS, for example, average execution, load balancing, and makespan. In the proposed algorithm, tasks are executed on nodes according to their priority. The highest priority task will be executed first. In [20] author tries to improve the network bottleneck using a proxy network. Through smart routing, the author proves that network bottleneck can be reduced, and workflow application performance increases significantly. The novelty of this work is that multiple proxies are included during the workflow optimization stages with different performance matrices.

Based on the three main stages of workflow application, the cost optimization model is presented in [21]. The cost parameters are based on pre-scheduling, during scheduling, and after scheduling. The

author also considers the different number of VMs in the experimental environment. The author in [22] presented a serverless optimized container scheduling technique for a data-intensive edge environment. This methodology makes trade-offs between computational movement and data and considers the workflow-related requirements. This scheduling considers the execution time minimization, cloud execution cost minimization, and uplink usage constrained.

In this article, we proposed a meta-heuristic technique to minimize the workflow makespan under the workflow deadline constraint in an edge environment. Because we noticed that none of the existing methodologies considers the multiple service providers' environment in the edge computing system. For this reason, in the proposed methodology, we considered the multiple service providers, heterogeneous resources, performance variation of computing instances, dynamic acquisition, and transmission delay time in the proposed algorithm.

3. Problem statement and Methodology

3.1 . Problem statement

Scientific applications demand high quality of quick service that should be provided to their tasks. Simply scheduling workflow tasks on the edge server may increase the makespan in heterogeneous and performance variant multi-service provider's environment. Therefore, heterogeneous and performance variant edge servers offered by multiple service providers are considered in the presented work. Only those servers will be selected, which can help to minimize the overall workflow makespan and meet the workflow requirements. Figure 1, illustrated the example of workflow. User devices are linked to access point (AP) via the internet i.e., 3G, 4G, 5 G, Wi-Fi, and APs possess the computational resources deployed by various service providers. In our scenario, three different kinds of service providers are considered, sp_1 , sp_2 and sp_3 .The heterogeneity and performance aware workflow makespan minimization in edge environment problem can be formulated as follow.

3.2 Methodology

Multiple service providers with heterogeneous and performance variant resources in the edge environment offer fast services to the clients. Computational resources can be denoted as a set of servers $S = \{s_{11}, s_{12}, \mathrm{K}\ , s_{1n}, \mathrm{K}\ , s_{21}, s_{22}, \mathrm{K}\ , s_{2n}, \mathrm{K}\ , s_{m1}, s_{m2}, \mathrm{K}\ , s_{mn}\}$, where n denoted the server's ID and m denotes the service provider's ID. Therefore, the accessible servers $S = \{s_1, s_2, \mathrm{K}\ , s_k\}$ are available to the end user with the heteroginouse diverse attrcibutes $A_s = (R, C)$. Resources avaliable on server are denoted as R and capacity of the server is denoted as C . Each server has a set of limited resources which includes

$R = \{cpu(com\_capacity), memory, bandwidth\}$. Edge users can submit any computational requested which could be complex workflow. A scientific workflow is represented by a Directed Acyclic Graph (DAG) $W = (\tau, e)$, where $\tau = \{\tau_1, \tau_2, \text{K}, \tau_z\}$ denotes the set of tasks and e the set of edges. τ_1 and τ_z are the entry and exit tasks respectively. In case of no single entry or exit task, a dummy entry/exist task with zero execution time can be added. An edge $(\tau_x, \tau_y) \in \tau$ indicates the control and data dependency from task τ_x to τ_y. A sample DAG graph is shown in Figure 1 with different resource demands. Each workflow has a set of attributes $A_{workflow} = (Data, Deadline, bandwidth)$. Deadline is the workflow constraint which is usually expressed in the SLA documents. It is obvious that user's device and server has the different bandwidth, keeping in mind that user and server have the different data sending/ receiving data rate. Data transfer time $Transf\_Time_{\tau_i}$ can be estimated as sending data $Data_{\tau_i}$ of task divided by bandwidth BW as shown in (1).

$$Transf\_Time_{\tau_i} = \frac{Data_{\tau_i}}{BW} \quad (1)$$

Scientific workflow consists z number of interdependent tasks. According to (2), $Exe\_time_{\tau_i}$ is the maximum execution time of task τ_i on edge server s_k and the data size $Data_{\tau_i}$ of task τ_i. $com\_capacity_{s_k}$ indicates the computational capacity of edge server s_k. $Exe\_time_{\tau_i}$ can be estimated as the data $Data_{\tau_i}$ size of task divided by the processing capacity of the server $com\_capacity_{s_k}$ in terms of Floating-Point Operation Per Second (FLOPS).

$$Exe\_time_{\tau_i} = \frac{Data_{\tau_i}}{com\_capacity_{s_k}}, \ \tau_i \in \tau \quad (2)$$

In (3), $Exe\_time''_{\tau_j}$ denotes the expected remaining execution time of currently running task τ_j on edge server s_k. It needs to be known in case the computational resource is busy in executing other task τ_j. $Exe\_time''_{\tau_j}$ can be estimated in a similar way as (2).

$$Exe\_time''_{\tau_j} = \frac{Data''_{\tau_j}}{com\_capacity_{s_k}} \quad (3)$$

Earliest starting time of task τ_i on the computational resource can be calculated as follow

$$EST_{\tau_i} = Exe\_time''_{\tau_j} + Transf\_Time_{\tau_i} \quad (4)$$

Same as the earliest start time of the task τ_i on computation resource s_k, earliest finish time is task τ_i can also be calculated as illustrated in (6). $Total\_Time_{\tau_i}$ represent the overall execution and transfer time of the task to resource as denoted in (5).

$$Total\_Time_{\tau_i} = Transf\_Time_{\tau_i} + Exe\_time_{\tau_i} \quad (5)$$

$$EFT_{\tau_i} = Exe\_time''^{s_k}_{\tau_j} + Exe\_time^{s_k}_{\tau_i} + Total\_Time_{\tau_i} \quad (6)$$

According to (7), over all expected workflow completion time can be calculated as

$$Total\_Time_{workflow} = \left\{ Total\_Time_{\tau_1} + Total\_Time_{\tau_2} + \mathrm{K} + Total\_Time_{\tau_\eta} \right\} \quad (7)$$

In the proposed method, each task has deadline constraint which must be met. Since, each task is computed in a timely manner, and then the overall workflow can meet its deadline. To meet the workflow deadline constraint the total time and earliest starting time of task must be less than the sub-deadline, which is illustrated in (8).

$$Total\_Time_{\tau_i} \,\&\, EST_{\tau_i} < Deadline_{\tau_i} \quad (8)$$

Every server contains number of resources to fulfill the requirement of the end user. Resource availability can be acknowledged according to (9) and required available on the server can fulfill the necessities of the user can be estimated as in (10).

$$Resource_{s_k} = \begin{cases} 1 & \text{if } resource \text{ is available on } s_k \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

$$Resource\_req_{\tau_i} \le Resource\_avail_{s_k} \quad (10)$$

According to (9), $Resource_{s_k}$ is zero if there is resource s_k is available otherwise value is one. Server can only fulfill the demand of the user if the server has the enough capacity as illustrated in (11) demand capacity constraint. Therefore, algorithm will check the task's resource demand. It must be less than the server capacity in case of task offloading otherwise task will be moved to some other server where server have the enough capacity.

$$Demand_{\tau_i} \le Capacity_{s_k} \quad (11)$$

To solve the problem in a genetic manner, it is necessary to select the best and fittest individuals to produce the superior generation. These individuals are allowed to pass their genes to the next generation. Fitness function helps to determine the better and fittest individuals. In our scenario the fitness function is denoted as follow

$$Workflow\_Time = \sum_{x=1}^{n} Exe\_time_{\tau_i}^{s_k} + \sum_{x,j=1}^{\theta} min(Total\_Time_{\tau_i}^{s_k} + Exe\_time''^{s_k}_{\tau_j}) \quad (12)$$

$$Minimum(Workflow\_Time) \le Deadline_{Workflow} \quad (13)$$

4. Genetic Algorithm for Workflow Make-Span Minimization in Edge Computing Environment (GA-WMM)

Meta-heuristic algorithms are an evolutionary way to solve the NP-hard problems [50] instead of simple and straightforward ones in solving any sizeable problem. Existing studies [23-25] undoubtedly suggest the advantages of genetic algorithms' (GAs) efficiency for workflow scheduling over the heuristic algorithms, i.e., Ant colony optimization and Particle swarm optimization. GA is highly used for the best

quality solutions for searching and optimization problems by using bio-inspired operations, e.g., selection, crossover, and mutation. Best parents are selected to participate in the production of the next generation from the population, where each individual participates. Two or more parents participate during the crossover operation to produce new offspring which can perform better than their insisters. In the mutation operation, random changes are made to enhance the performance of offspring so that offspring can perform better. Therefore, each new generation is highly improved and well-fit to their environment. Solutions are usually denoted in binary strings, e.g., 0s and 1s, but another encoding schemed is also possible, as presented in our method. Based on the problem description, we have considered all the genetic algorithm operations, such as initialization of population, selection, encoding, crossover, and mutation, in the proposed technique.

4.1. Initialization and Selection

The population is randomly generated during the initialization phase from a few individuals to hundreds. Each individual must fulfill the fitness value, and those individuals are dropped from the competition to produce the next generation that can not fulfill the fitness value. Encoding sequence should consider the parent-child relationship of workflow tasks; this means that no task can start its execution until the predecessors of the task are not executed. Two steps are followed to generate the population: i) z number of workflow tasks order list l is generated as in (14); ii) Servers with the earliest availability and lowest transmission time delay with better performance variation are selected for the task's allocation.

$$\mathrm{l}_i = \{\mathrm{l}_1, \mathrm{l}_2, \mathrm{l}_3, \mathrm{K}, \mathrm{l}_\theta\} \in \mathrm{l} \quad (14)$$

To select the best and fittest genomes for the next generation competition-based selection is performed. Only those genomes are selected which can satisfied the requirements of the workflow task and have enough resources to accommodate the task. To produce the near optimize solution proposed algorithm only select the most suitable server to schedule the workflow task.

4.2. Encoding

Four vectors are encoded to present the genome of workflow task allocation problem in the encoding operation. (i) Task ID, (ii) Provider's ID, (iii) Task to Server, and (iv) Weight to server. In the beginning, "Task ID" is assigned as an integer value to each workflow task considering the precedence condition. Following this, in "Provider's ID" encoding represents the service providers ID. In our scenario 3 service providers are offering their heterogeneous resources. Therefore, the "Provider's ID" integer values will be from 1 to 3. In the third vector, "Task to Server", encoding is presented in which the server is selected according to the requirements of the workflow task. This server may belong to any service provider. Last vector represents the weight assigned to servers according to their performance and efficiency. If the edge server has better efficiency and high performance, it will have the lowest weighted value in the vector. Server with higher weighted value in the vector has the lowest performance than others servers.

A complete encoding model of Figure 1 is presented in Table 1. In the initial step of encoding, order of the workflow task is depending on the precedence constraint in which parent child relationship must be considered. According to the precedence constraint integer value is assigned to each task of workflow. Vector length is depended on the number of tasks in the workflow which is from task τ_1 to task τ_z. Each service provider has the heterogeneous servers with different resources. These service providers are assigned a vector as "Provider's ID". Every server is selected according to fitness values than workflow task is scheduled on the selected server. Vector three "Task to server" represents the mapping of the workflow tasks.

Table 1 Encoding scheme

	τ_1	τ_4	τ_2	τ_3	τ_6	τ_5	τ_7
Task ID	1	2	3	4	5	6	7
Provider's ID	2	1	1	3	2	2	1
Task to Server	**2**	**1**	**1**	**2**	**1**	**3**	**1**
Weight to Server	**0.3**	**0.1**	**0.1**	**0.1**	**0.3**	**0.2**	**0.1**

4.3 . Crossover

Crossover operation can create new offspring by combining two or more best-selected genomes/individuals. Newly produced offspring are expected to be better and fittest than their insisters. The probability Pc [0, 1] is used to rearrange the new individuals in the current population, and some individuals are merged with the existing individuals. In the genetic crossover operation, each workflow task is scheduled by considering the task requirements and constraints. Therefore, the selected computing instance can meet the requirements of workflow tasks and the deadline constraints. In the proposed work, two points, "a" and "b" from "parent 1" and "parent 2" are randomly selected in the crossover operation. For instance, a= 4 and b =6 from parent 1 and a=3 and b=5 from parent 2. Subsequently, algorithm will select the tasks τ_5, τ_6, τ_3 from parent 2 for offspring 1 and tasks τ_4, τ_6, τ_5 from parent 1 for the offspring 2 without changing the workflow task's order in the vector as show in subset of Figure 2.

In the next step these selected tasks are replaced with all their information in the parent 1 and

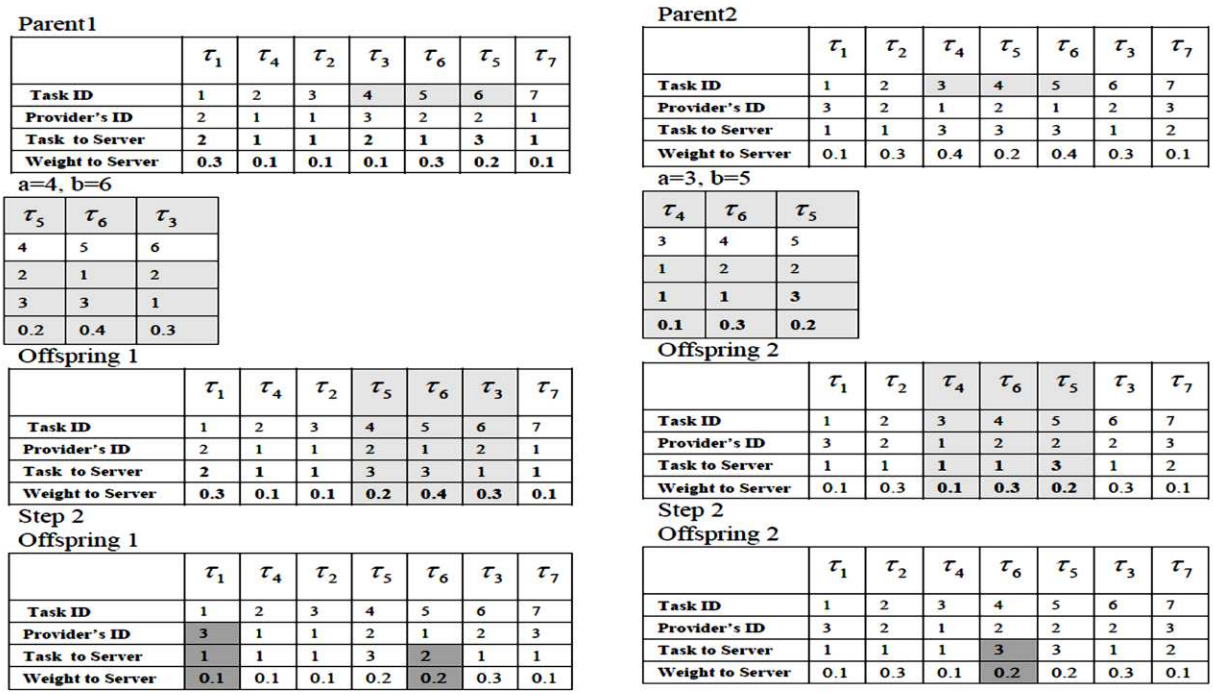

Figure 2 Crossover operation

parent 2 to produce the offspring. The other information includes "Providers ID", "Task to Server", and "Weight to Server" except the "Task ID" which will be the same. In the crossover operation's last step, task is scheduled to that server which can help to minimize the overall makespan by considering the deadline constraint of the workflow. This selected computational instant can fulfill all the requirements of the task within reasonable time.

4.3. Mutation

Mutation operation keeps the workflow tasks in order and follows the precedence constraints. It is compulsory to keep the order of tasks. Slide randomness into chromosomes during mutation operation can help to achieve the better results. Proposed algorithm has exchange and mutation similar to existing studies. During exchange mutation server, instance is randomly selected and two tasks are swapped. On the other hand, in the replace mutation, vacant instance from the population individual is re-assigned to the task. A random value is generated between 0 and 1 to use the mutation probability P_z, P [0, 1]. Replace mutation operation is performed if the value is greater than P_z, otherwise, exchange mutation operation is performed.

Table 2 Before Mutation operation

	τ_1	τ_4	τ_2	τ_3	τ_6	τ_5	τ_7
Task ID	1	2	3	4	5	6	7
Provider's ID	2	1	1	3	2	2	1
Task to Server	**2**	**1**	**1**	**2**	**1**	**3**	**1**
Weight to Server	**0.3**	**0.1**	**0.1**	**0.1**	**0.3**	**0.2**	**0.1**

Table 3 After Mutation operation

	τ_1	τ_3	τ_2	τ_4	τ_6	τ_5	τ_7
Task ID	1	4	3	2	5	6	7
Provider's ID	2	3	1	1	2	2	1
Task to Server	**2**	**2**	**1**	**1**	**1**	**3**	**1**
Weight to Server	**0.3**	**0.1**	**0.1**	**0.1**	**0.3**	**0.2**	**0.1**

5. Experiment and Results

Intensive simulation is performed to evaluate the performance of a proposed meta-heuristic algorithm. It is noticeable that the proposed algorithm outperforms the other competitor algorithms.

5.1. Experimental Setup

Evaluation of the proposed algorithm is carried out by using Huawei and Tencent rational workflow data, and results are analyzed with baseline algorithms. The workflow makespan is examined with different sizes of workflows which consists of a dissimilar number of interdependent tasks. It is supposed that any server on edge can fulfill the workflow requirements; however, servers are physically heterogeneous and have performance variations. For the performance evolution of the proposed algorithm deadline constraint is defined for each workflow. Single Hard-Deadline constraint is used for each workflow to estimate the deadline of the workflow. In (15), CT is the workflow completion time and f is the workflow deadline factor. We kept the f value small to keep it generate the hard deadline value.

$$Deadline = CT \times (0.5 + f) \quad (15)$$
$$0 \le f \le 1$$

5.2. Competitor Algorithms

The main differences between our work and baseline uniform algorithm are heterogeneity, performance variation, acquisition, and transmission delay. This algorithm ignores the dynamic characteristics and performance variations of the edge computing environment. In addition, when mapping tasks to the required instances, the acquisition, and transmission delay have a significant impact on the makespan of the entire workflow. In uniform algorithm acquisition and transmission, the delay is fixed. However, the acquisition delay and data transmission time are dynamic in the proposed algorithm. According to the service provider, the instance performance variations, acquisition delay, and transmission delay of the instance are considered in the proposed work.

Our second competitive algorithm is random, and the acquisition and transmission delays are considered to be dynamic. However, the algorithm randomly chooses any server for mapping the workflow task regardless of heterogeneity, performance variation, the acquisition delay, and data transmission time factors.

.

5.3. Results and Analysis with Competitor Algorithms

The core objective of our proposed method is to reduce the makespan of the scientific workflow by minimizing the delays in the multi-service provider environment.

According to Figure 3 and Figure 4, our method clearly outperforms other competitor algorithms in terms of measured workflow makespan for both Huawei and Tencent data. This is because the proposed algorithm only selects the instance which has the lowest execution time, transmission, acquisition delay, and higher performance than other edge servers. In our experiment, workflow execution with different number of tasks is performed ten times. Graphs in Figure 3 and Figure 4 show the total number of tasks in the X-axis and makespan of the workflow in the Y-axis. Proposed GA-WMM better than the Random method but extremely better than the Uniform technique. As shown by Figure 5 and Figure 6, our proposed method achieves lower total acquisition and transmission delay as well. Graphs in Figure 5 and Figure 6 show the total number of workflow tasks on the X-axis and the delay time of the workflow on the Y-axis. Proposed GA-WMM performs slightly better than random when the workflow size is small, but as the workflow size increases, the proposed GA-WMM outperforms its competitor methods. Figure 5 and Figure 6 prove that delay factors cannot be ignored in a real-edge environment to achieve real-

time service. These delay factors play an important role in the workflow makespan. Therefore, if the delay factors are minimized, the overall makespan will also be reduced. Table 4 represents the percentage values of the deadline success rate of each algorithm.

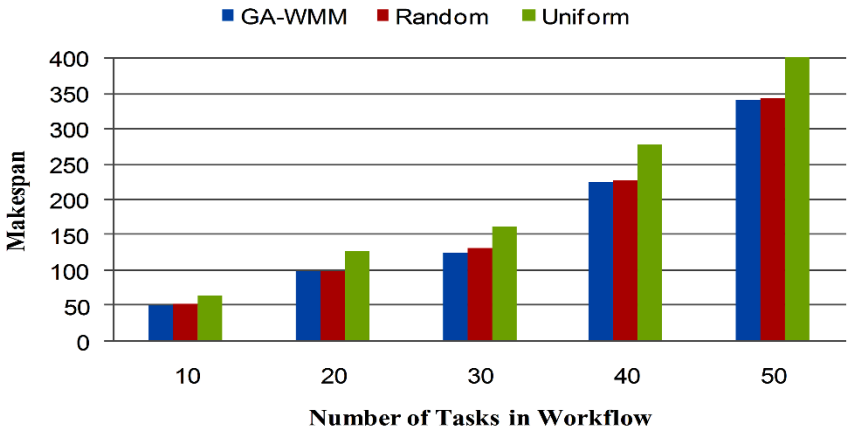

Figure 3 Makespan of Huawei workflow data.

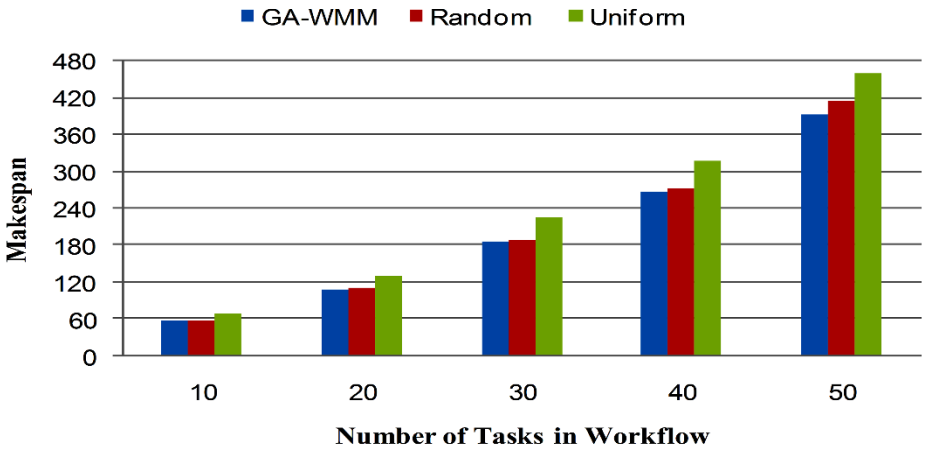

Figure 4 Makespan of Tencent workflow data.

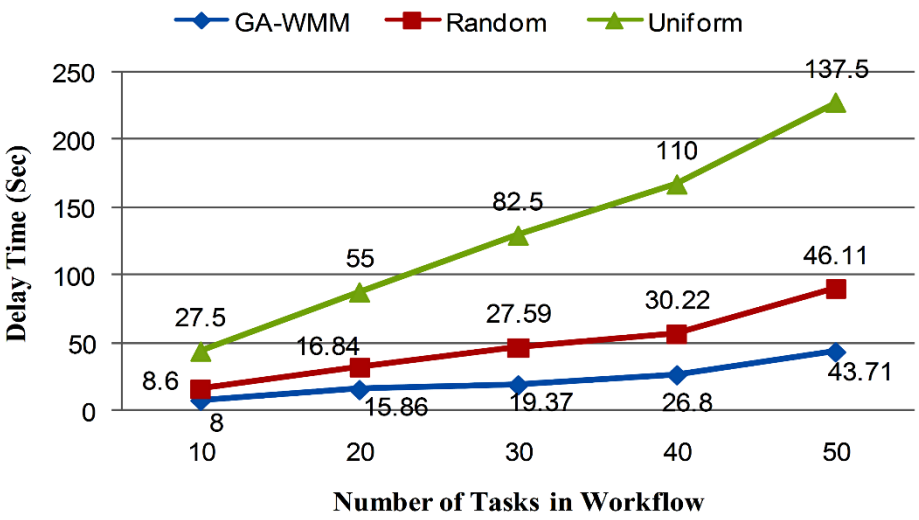

Figure 5 Delay time in Huawei workflow data.

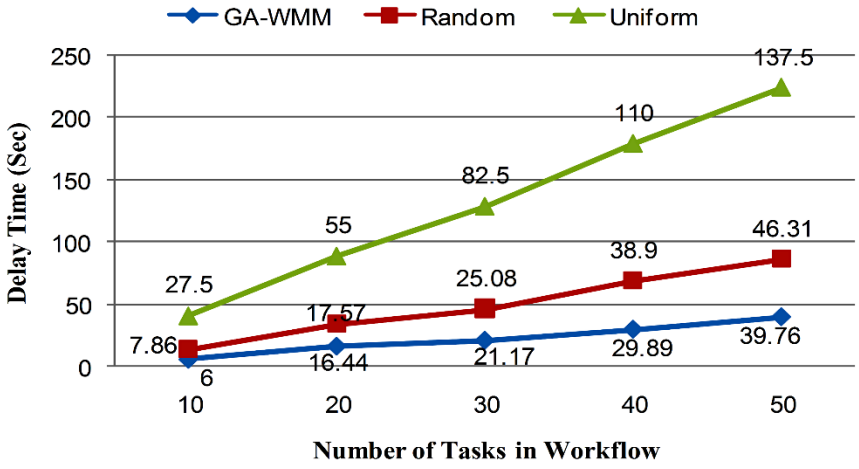

Figure 6 Delay time in Tencent workflow data.

Table 4 Deadline constraints success rate

Algorithms	Success Rate (%)	Percentage Improvement of proposed Algorithm
Random	94.6	5.4
Uniform	89.9	10.1
GA-WMM	100	-

Our experiment executed the workflow of size 50 and the number of service providers from 3 to 7, as shown in Figure 7. We noticed that with every number of the total service provider, the makespan is different. This is because the proposed algorithm tries to select the instances from the population with the lowest execution time, transmission, acquisition delay, and higher performance than other edge servers. So, if we increase the number of instances, we will increase the probability that the algorithm will find the desired instance which can fulfill the QoS requirements. Nonetheless, this does not mean that we will achieve the lowest makespan if we increase the service provider or instances. Because maybe those instances have much higher delay times, as clearly shown in Figure 7 in some cases.

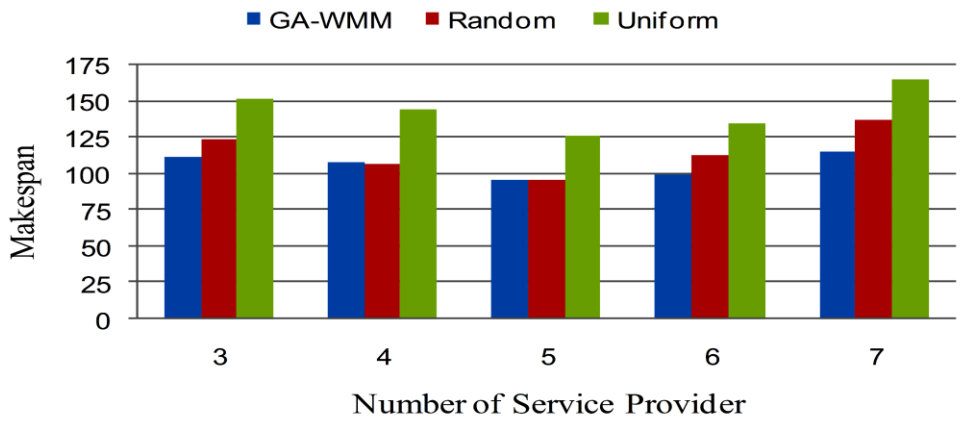

Figure 7 Makespan with different number service providers.

In this experiment, we changed the number of service providers up to 7, as shown in Figure 7. We noticed that with every number of the service provider, the makespan is dissimilar. This is because the proposed algorithm tries to select the instances from the population with the lowest execution time, transmission, acquisition delay, and higher performance than other edge servers.

6. Conclusion and Future Work

In this paper, a meta-heuristic-based algorithm is presented to schedule the scientific workflow on the edge servers in multiple service providers' environments. Our proposed algorithm aims to generate appropriate scheduling and provisioning choices to minimize the makespan of scientific workflow under deadline constraints. The proposed algorithm considers the acquisition delay, heterogeneity, transmission delay time, and performance variation of edge servers. Extensive case studies have been carried out in the real-world commercial clouds, i.e., Tencent and Huawei clouds. The presented

technique delivers categorically better results than the other competitor algorithms on the premise of satisfying the QoS constraints.

In the context of edge computing, it is known to us that minor computational problems can be solved on the edge servers, and cloud systems can also participate, where higher computational is required. However, to the best of our knowledge, there is no existing study available about the cooperation of edge servers for higher computational demanding tasks in multiple service providers' environments. In the future, we will try to explore the research area where more computationally demanding tasks can be executed on the edge server (servers offered by the multiple service providers) by collaborating instead of sending these tasks to cloud systems.

Funding: Not applicable.
Conflicts of Interest: The authors declare that there is no conflict of interest.
Availability of data and material: Not applicable.
Code availability: At this stage, we can not provide the code because we are extending our project.
Authors' contributions: All authors contributed equally.

References

[1] M. Farid, R. Latip, M. Hussin, and N. A. W. J. S. Abdul Hamid, "A survey on QoS requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing," vol. 12, no. 4, p. 551, 2020.

[2] S. S. H. Bukhari and Y. J. I. J. o. W. S. R. Xia, "A Novel Completion-Time-Minimization Scheduling Approach of Scientific Workflows Over Heterogeneous Cloud Computing Systems," vol. 16, no. 4, pp. 1-20, 2019.

[3] S. Banerjee, M. Adhikari, S. Kar, U. J. A. J. f. S. Biswas, and Engineering, "Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud," vol. 40, no. 5, pp. 1409-1425, 2015.

[4] S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar, R. J. J. o. N. Buyya, and C. Applications, "SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter," vol. 45, pp. 108-120, 2014.

[5] P. Wang, Y. Lei, P. R. Agbedanu, and Z. J. I. A. Zhang, "Makespan-driven workflow scheduling in clouds using immune-based PSO algorithm," vol. 8, pp. 29281-29290, 2020.

[6] R. Zhang and W. Shi, "Research on Workflow Task Scheduling Strategy in Edge Computer Environment," in *Journal of Physics: Conference Series*, 2021, vol. 1744, no. 3, p. 032215: IOP Publishing.

[7] J. K. Konjaang and L. J. J. o. C. C. Xu, "Multi-objective workflow optimization strategy (MOWOS) for cloud computing," vol. 10, no. 1, pp. 1-19, 2021.

[8] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, and B. J. I. T. o. S. C. Luo, "Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds," vol. 11, no. 4, pp. 713-726, 2015.

[9] Y. Chawla, M. J. I. J. o. E. T. Bhonsle, and T. i. C. Science, "A study on scheduling methods in cloud computing," vol. 1, no. 3, pp. 12-17, 2012.

[10] Y. Lin, H. J. I. T. o. P. Shen, and D. Systems, "CloudFog: leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service," vol. 28, no. 2, pp. 431-445, 2017.

[11] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. J. I. T. o. E. T. i. C. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," vol. 5, no. 1, pp. 108-119, 2017.

[12] A. Mukherjee, D. De, and D. G. J. I. T. o. C. C. Roy, "A power and latency aware cloudlet selection strategy for multi-cloudlet environment," 2016.

[13] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. J. I. I. o. T. J. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," vol. 3, no. 6, pp. 1171-1181, 2016.

[14] M. Jia, J. Cao, and W. J. I. T. o. C. C. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," vol. 5, no. 4, pp. 725-737, 2017.

[15] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. J. I. I. o. T. J. Han, "Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining Stackelberg game and matching," vol. 4, no. 5, pp. 1204-1215, 2017.

[16] I. Paik, Y. Ishizuka, Q.-M. Do, and W. Chen, "On-Line Cost-Aware Workflow Allocation in Heterogeneous Computing Environments," in *2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, 2018, pp. 209-216: IEEE.

[17] H. Derhamy, M. Andersson, J. Eliasson, and J. Delsing, "Workflow management for edge driven manufacturing systems," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, 2018, pp. 774-779: IEEE.

[18] I. Al Ridhawi, Y. Kotb, and Y. J. I. A. Al Ridhawi, "Workflow-net based service composition using mobile edge nodes," vol. 5, pp. 23719-23735, 2017.
[19] X. Wu, M. Deng, R. Zhang, B. Zeng, and S. J. P. C. S. Zhou, "A task scheduling algorithm based on QoS-driven in cloud computing," vol. 17, pp. 1162-1169, 2013.
[20] S. Ramakrishnan, R. Reutiman, A. Chandra, and J. Weissman, "Accelerating distributed workflows with edge resources," in *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*, 2013, pp. 2129-2138: IEEE.
[21] E. N. Al-Khanak *et al.*, "A Heuristics-Based Cost Model for Scientific Workflow Scheduling in Cloud," vol. 67, no. 3, 2021.
[22] T. Rausch, A. Rashed, and S. J. F. G. C. S. Dustdar, "Optimized container scheduling for data-intensive serverless edge computing," vol. 114, pp. 259-271, 2021.
[23] J. Meena, M. Kumar, and M. J. I. A. Vardhan, "Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint," vol. 4, pp. 5065-5082, 2016.
[24] Z. Zhu, G. Zhang, M. Li, X. J. I. T. o. p. Liu, and d. Systems, "Evolutionary multi-objective workflow scheduling in cloud," vol. 27, no. 5, pp. 1344-1357, 2016.
[25] Z.-G. Chen, K.-J. Du, Z.-H. Zhan, and J. Zhang, "Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, 2015, pp. 708-714: IEEE.

Biography

Syed Sabahat Hussain Bukhari received the PhD. degree in Software Engineering from Chongqing University, Chongqing, P. R. China in 2019 and Master degree in Computer Science from COMSATS University Islamabad, Pakistan in 2011. Currently, He is working in Neijiang Normal University, Sichuan, P. R. China. Sabahat's research interests are in the areas of cloud computing, edge computing, Real-time system, distributed and parallel computing.

Muhammad Usman Younus received his Doctorate and master's degree in engineering from the University of Toulouse (III) Paul Sabatier France and University of Engineering and Technology Lahore Pakistan 2020 & 2014, respectively. His research interests include Wireless Sensor Network, Software Defined Networking, Energy Optimization, Wireless Communication, and Machine Learning. He has more than twenty international journal and conference publications. He is a member of PEC, IEEE, etc. and reviewer of some journals and conferences.

Zain ul Abidin Jaffri received his M.E. degree in Electronics and Communication Engineering and Ph.D. in Communication and Information Systems from Chongqing University, Chongqing, China. He is currently working as an Associate Professor in the College of Physics and Electronic Information Engineering, Neijiang Normal University, Neijiang, China. His main research areas include Wireless Sensor Networks, Antenna Design and Propagation, mobile and wireless communication networks, and

Advanced Computer Networks.

Muhammad Arshad Shehzad Hassan received the Ph.D. degree in Electrical Engineering from Chongqing University, Chongqing, China, in 2019. He is currently an Assistant Professor with The University of Faisalabad, Faisalabad, Pakistan. His research interests include intelligent modeling and control, optimization of microgrids, microgrid control, and transmission line fault location and protections.

Muhammad Rizwan Anjum received his Ph.D degree from Beijing Institute of Technology, Beijing China in 2015. M. Engg. in Telecommunication and Control Engineering and B.Engg. in Electronic Engineering in 2011 & 2007 respectively from Mehran UET Jamshoro, Pakistan. Presently working as Associate Professor in the Department of Electronic Engineering, The Islamia University of Bahawalpur, Pakistan. He has more than 25 international conferences and journal publications. He is a member of PEC, IEEEP, IEP, IJPE, UACEE, IACSIT, ICCTD, IACSIT, IAENG, etc. and reviewer of several journals and conferences.

Sanam Narejo is currently working as Assistant professor in Department of Computer Systems Engineering, Mehran University of Engineering and Technology (MUET), Jamshoro. She has completed her PhD from Politecnico Di Torino, Italy in 2018. She received her Masters degree in Communication Systems and Networking from MUET. Her research interests include Signal and Image Processing, Machine Learning and Deep Learning Architectures. She has also been a member of Italian Society of Neural Networks (SIREN).

Figures

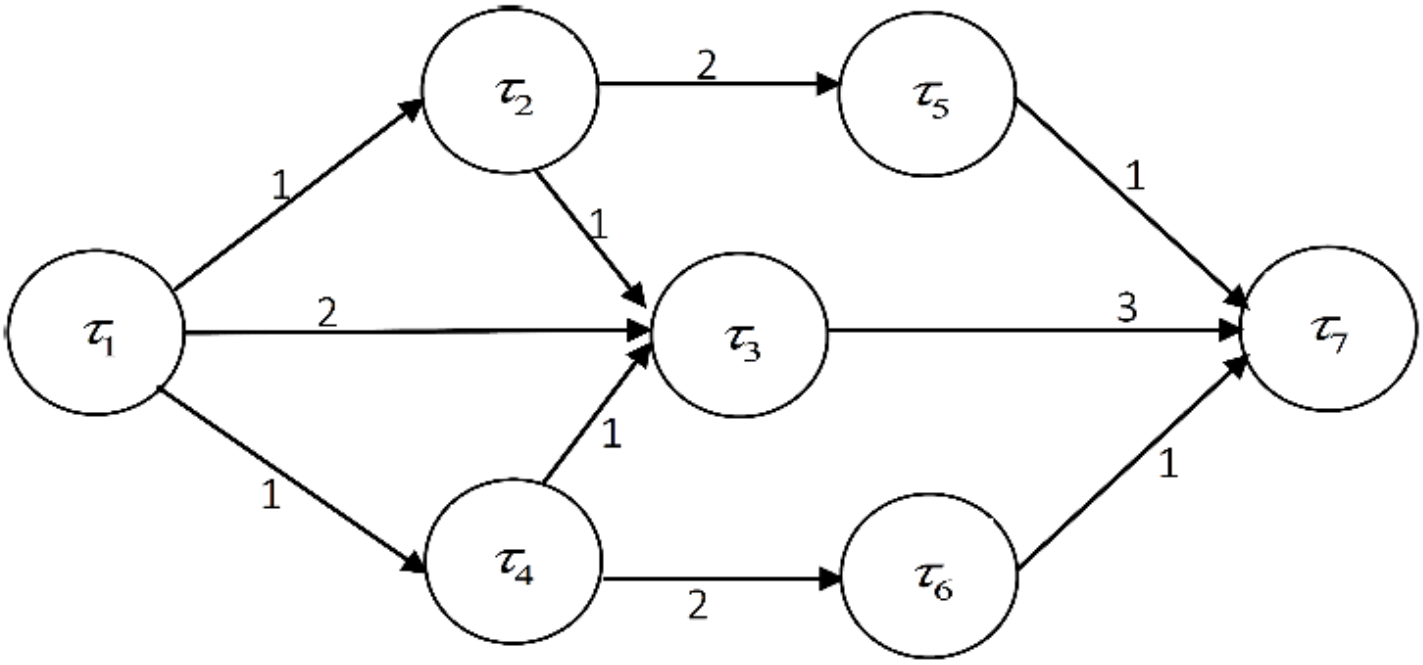

Figure 1

Example of workflow

Parent1

	τ_1	τ_4	τ_2	τ_3	τ_6	τ_5	τ_7
Task ID	1	2	3	4	5	6	7
Provider's ID	2	1	1	3	2	2	1
Task to Server	2	1	1	2	1	3	1
Weight to Server	0.3	0.1	0.1	0.1	0.3	0.2	0.1

a=4, b=6

τ_5	τ_6	τ_3
4	5	6
2	1	2
3	3	1
0.2	0.4	0.3

Offspring 1

	τ_1	τ_4	τ_2	τ_5	τ_6	τ_3	τ_7
Task ID	1	2	3	4	5	6	7
Provider's ID	2	1	1	2	1	2	1
Task to Server	2	1	1	3	3	1	1
Weight to Server	0.3	0.1	0.1	0.2	0.4	0.3	0.1

Step 2

Offspring 1

	τ_1	τ_4	τ_2	τ_5	τ_6	τ_3	τ_7
Task ID	1	2	3	4	5	6	7
Provider's ID	3	1	1	2	1	2	3
Task to Server	1	1	1	3	2	1	1
Weight to Server	0.1	0.1	0.1	0.2	0.2	0.3	0.1

Parent2

	τ_1	τ_2	τ_4	τ_5	τ_6	τ_3	τ_7
Task ID	1	2	3	4	5	6	7
Provider's ID	3	2	1	2	1	2	3
Task to Server	1	1	3	3	3	1	2
Weight to Server	0.1	0.3	0.4	0.2	0.4	0.3	0.1

a=3, b=5

τ_4	τ_6	τ_5
3	4	5
1	2	2
1	1	3
0.1	0.3	0.2

Offspring 2

	τ_1	τ_2	τ_4	τ_6	τ_5	τ_3	τ_7
Task ID	1	2	3	4	5	6	7
Provider's ID	3	2	1	2	2	2	3
Task to Server	1	1	1	1	3	1	2
Weight to Server	0.1	0.3	0.1	0.3	0.2	0.3	0.1

Step 2

Offspring 2

	τ_1	τ_2	τ_4	τ_6	τ_5	τ_3	τ_7
Task ID	1	2	3	4	5	6	7
Provider's ID	3	2	1	2	2	2	3
Task to Server	1	1	1	3	3	1	2
Weight to Server	0.1	0.3	0.1	0.2	0.2	0.3	0.1

Figure 2

Crossover operation

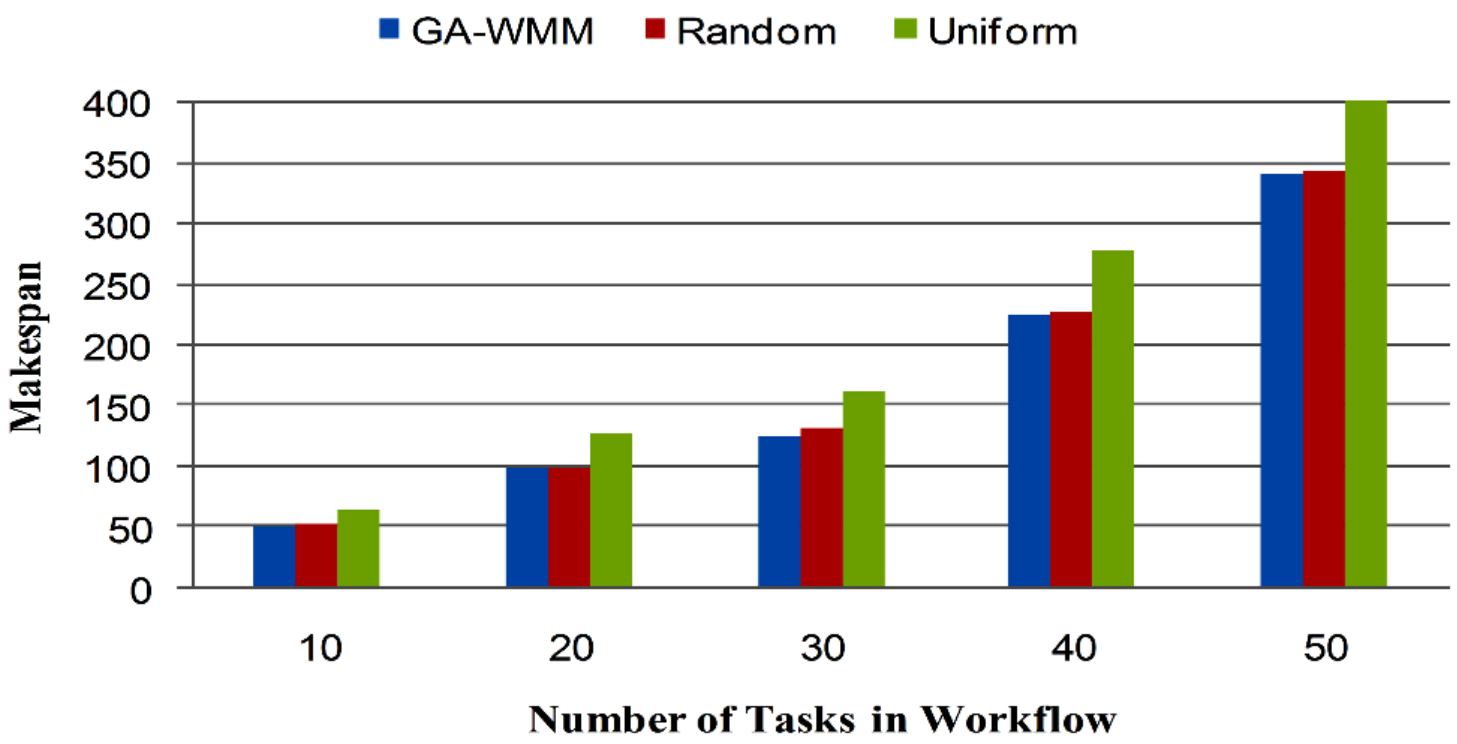

Figure 3

Makespan of Huawei workflow data.

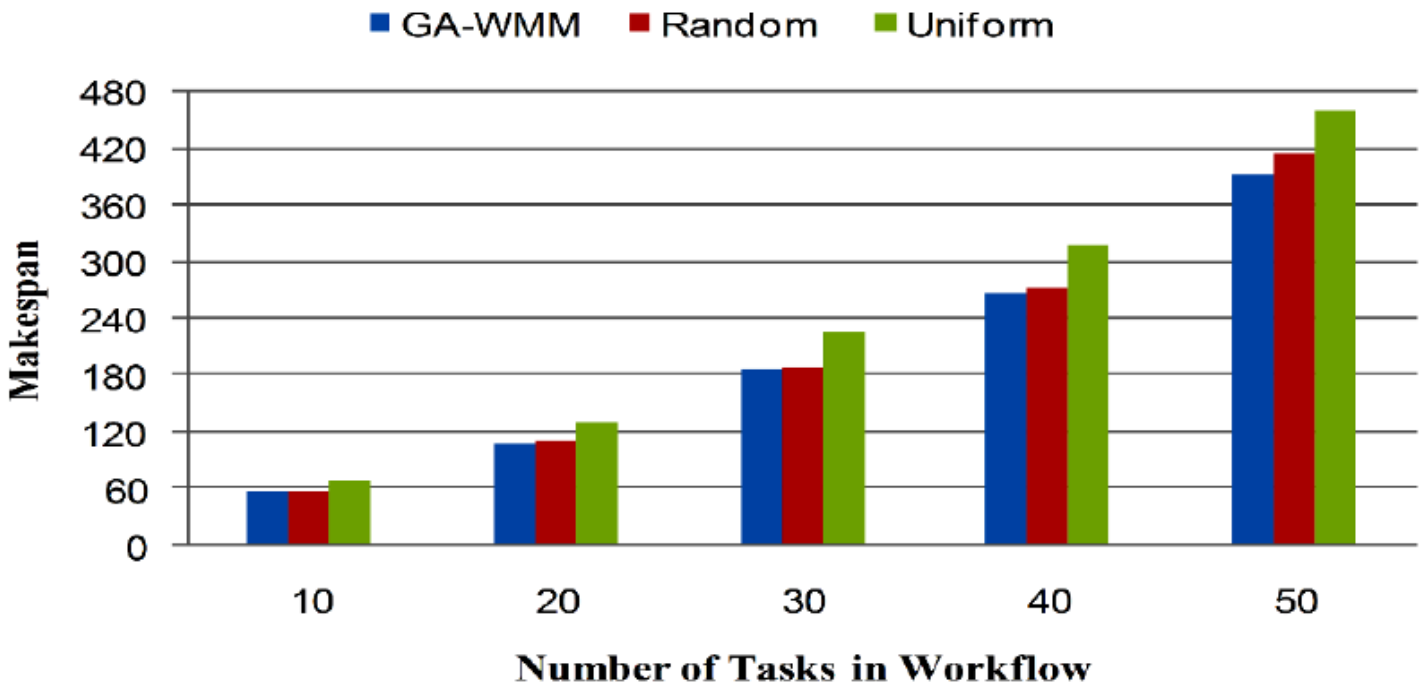

Figure 4

Makespan of Tencent workflow data.

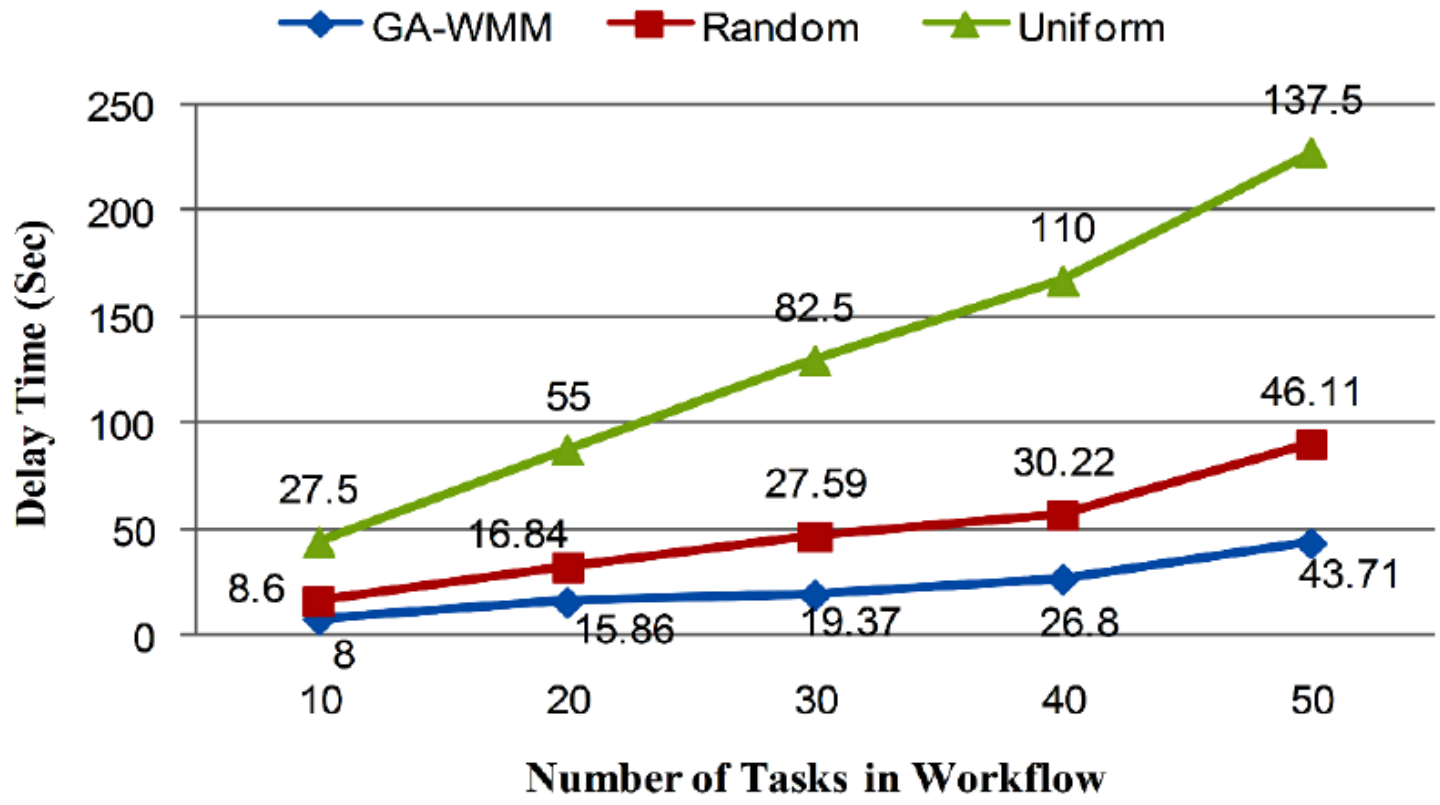

Figure 5

Delay time in Huawei workflow data.

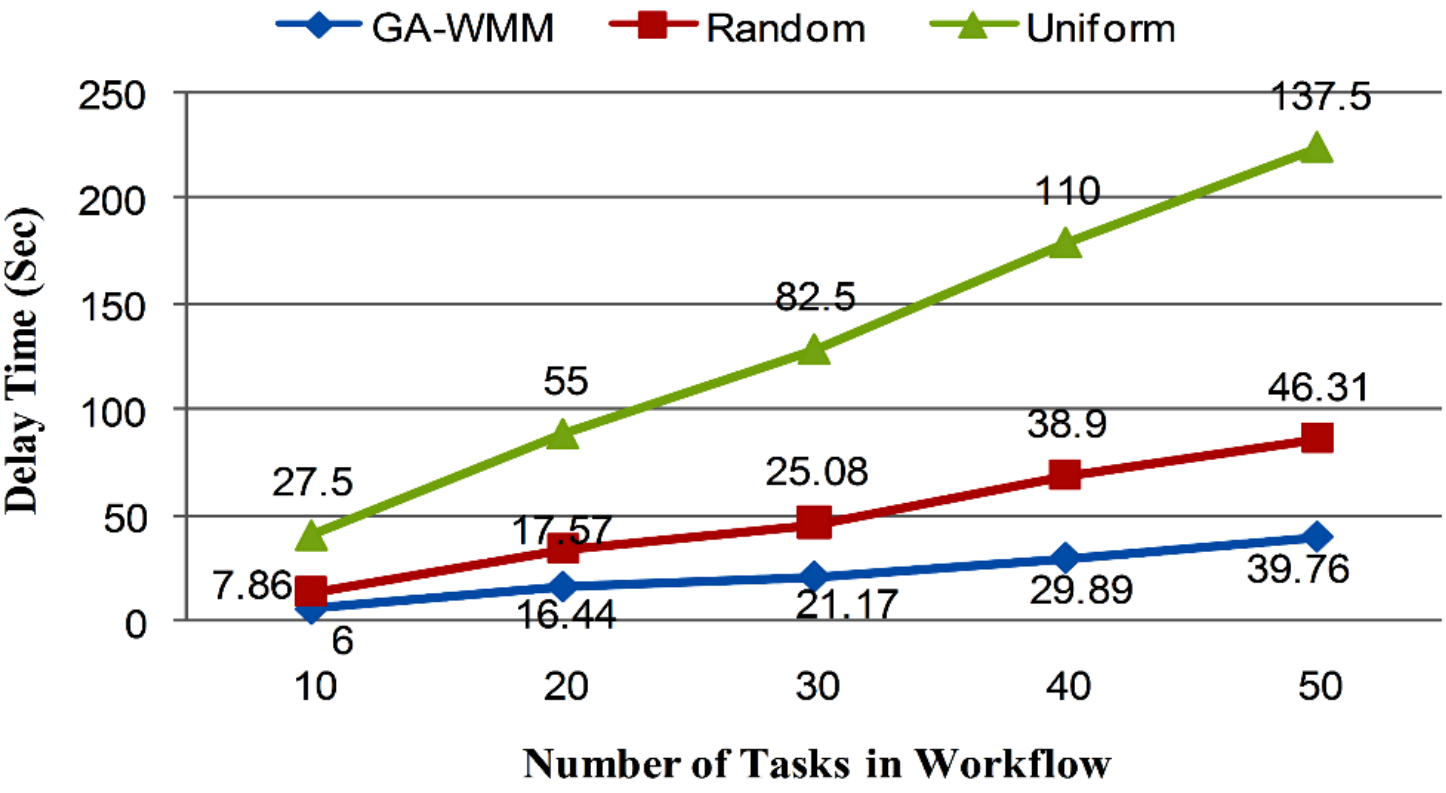

Figure 6

Delay time in Tencent workflow data.

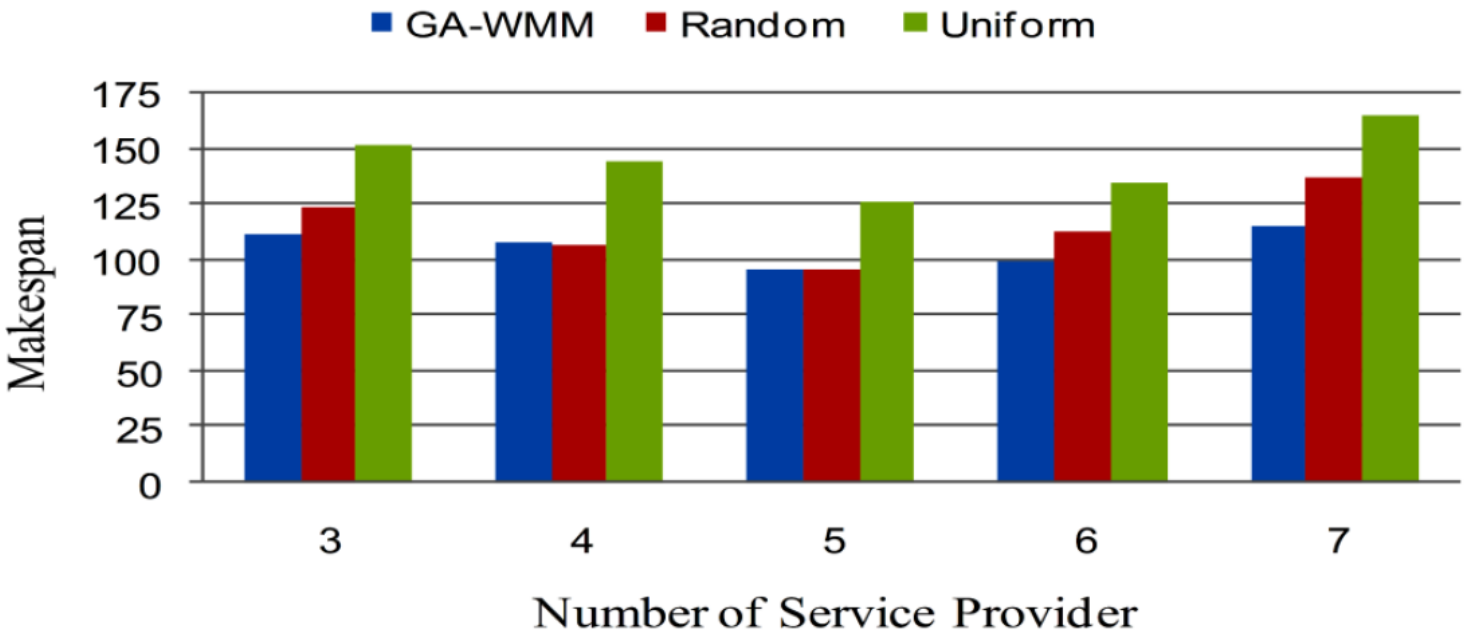

Figure 7

Makespan with different number service providers.