OPSEARCH
https://doi.org/10.1007/s12597-021-00508-6

APPLICATION ARTICLE

A hyper-heuristic selector algorithm for cloud computing scheduling based on workflow features

Abdolreza Rasouli Kenari[1] · Mahboubeh Shamsi[1]

Accepted: 13 January 2021

Abstract

This study focuses on the presentation of a new algorithm for scheduling workflows on heterogeneous distributed systems such as cloud computing. Since heterogeneous distributed systems deal with different types of resources, scheduling of applications on cloud resources plays an important role in the computing environment. Due to being heterogeneous and dynamic properties of resources as well as large numbers of tasks with different characteristics and dependencies among tasks, scheduling tasks on cloud computing is referred to as an NP-hard problem. Heuristic methods are one of the common approaches to solve this problem. Heuristic algorithms according to the specifications of resources and workflow structure could be superior to the rule-based methods. However, it is difficult to define which heuristic algorithm is performed better than the rest. Therefore, the choice of appropriate heuristic algorithms based on the circumstances can be effective. Moreover, the hyper-heuristic algorithm obtains higher performance. In this study, a new method is presented to improve the Hyper-Heuristic Scheduling Algorithm for the cloud using the decision tree method to select a convenient heuristic algorithm based on the characteristics of resources and workflows by considering evaluation criteria such as cost and Makespan. Finally, the presented algorithm is evaluated by Workflowsim and using RapidMiner. The simulation results demonstrate that our proposed algorithm outperforms existing approaches in terms of Makespan and Accuracy.

Keywords Cloud computing · Workflow · Heuristic algorithm · Heterogeneous distributed systems · Decision tree · Makespan

✉ Abdolreza Rasouli Kenari
rasouli@qut.ac.ir

Mahboubeh Shamsi
shamsi@qut.ac.ir

[1] Faculty of Electrical and Computer Engineering, Qom University of Technology, Qom, Iran

1 Introduction

In recent years, enhancing the efficiency of information systems has been paid more attention to computation, analysis, and storage. Therefore, distributed systems and parallel computing are identified as the main technologies to enhance the performance of various computer systems by considering inherent limitations in different areas. To achieve these objectives, scheduling [1, 6, 21] should be considered as the main strategy regarding the constraints of different resources and tasks of applications.

With the rapid growth in the capability of computer technologies and the internet, cloud computing [2, 3, 9, 18] is developed as a novel paradigm of distributed computing for information systems. Besides, cloud computing is known as a suitable solution to provide users with high computation tasks. However, issues such as job scheduling on cloud computing systems are still NP-hard [10] problem. Although the rule-based algorithms are applied for task scheduling because of their easy and simple implementation, these algorithms are not efficient methods in the case of large scale or complex scheduling problems due to being their results far from the optimal solution. Therefore, an optimal solution must be provided for scheduling tasks on cloud resources.

Since heuristic algorithms are chosen as an effective way to solve NP-hard problems, using the heuristic algorithm is a promising method for scheduling problems [28] has attracted many researchers to employ the potential capabilities of these algorithms for scheduling.

Furthermore, the integration of two or more heuristic algorithms is applied to a single hybrid heuristic algorithm causing performance enhancement [28]. Besides, a hyper-heuristic algorithm is proposed to choose a set of several heuristic algorithms including simulated annealing [15], genetic algorithm [12], particle swarm optimization [14] and colony optimization [7] resulting in leveraging strength points of different heuristic algorithms [27].

The main aim of this paper is to find an efficient method for choosing a suitable algorithm from a set of candidate heuristic algorithms through decision making. Indeed, the decision tree is applied to select the proper heuristic algorithm based on the prepared database improving the hyper-scheduling algorithm in terms of Makespan and accuracy.

The main contribution of the paper is summarized as follows:

1. The proposed algorithm employs a decision tree to choose the proper heuristic algorithm from a set of heuristic candidates set for scheduling on cloud computing to reduce Makespan and to improve accuracy.
2. The proposed algorithm is dependent on the characteristics of input workflow making to select the proper heuristic algorithm through the prepared database.
3. The proposed algorithm ignores the selection of a heuristic algorithm for each iteration compared with the hyper-heuristic algorithm leading to limit the time complexity into the selection operator of a heuristic algorithm for the first time. Therefore, it causes to reduce Makespan.

4. Workflowsim [12] and RapidMiner [14] are used as the simulation tools to evaluate the performance of the proposed algorithm. Moreover, the simulation results are compared with the existing algorithms in terms of the reduction of Makespan as well as the enhancement of accuracy.

The rest of the paper is organized as follows: In Sect. 2, the scheduling problem is defined with an emphasis on cloud computing. Moreover, heuristic algorithms applied to schedule on the cloud are provided in Sect. 3. In Sect. 4, we present a new method for scheduling on cloud computing, and subsequently, an example is provided to explain a new method in Sect. 5. Finally, a conclusion is drawn in Sect. 6.

2 Definition of scheduling problem in cloud computing

A workflow is defined as a set of interdependent tasks modeled by directed acyclic graph (DAG) $w=(T, E)$ [22, 24] where T is the numbers of tasks and E is a set of directed edges. According to Fig. 1, each task is represented by node and dependency between task i and task j is represented by directed edge (e_i, e_j) between nodes i and j in which t_i and t_j are parent and child tasks, respectively. The child node could not be executed before executing parent nodes and providing the required information. For example, as shown in Fig. 1, task 3 will be started once its parent tasks 0 and 1 are completed. Moreover, each workflow should be executed. Besides, the starting and ending tasks, s and e, are considered as virtual tasks and should not be executed.

In addition to defining a workflow, the characteristics of a cloud service provider should be identified. Cloud service providers offer range of resources including virtual machines $VM=\{VM_1, VM_2,..., VM_\mathrm{m}\}$ with their specific characteristics including storage, network, bandwidth, speed, and cost. Scheduling problem [4, 19] here is defined as assigning a limited set of virtual machines $VM=\{VM_1, VM_2,..., VM_\mathrm{m}\}$ to a set of tasks $VM=\{T_1, T_2,..., T_n\}$ of each workflow by taking predefined constraints and measurements such as reducing the completion time of the last task (Makespan) and enhancing accuracy. Besides, Eq. 1 express scheduling problems on a cloud with emphasis *fcc* on the reduction of Makespan regarding the main constraints including overdue time and budget [22].

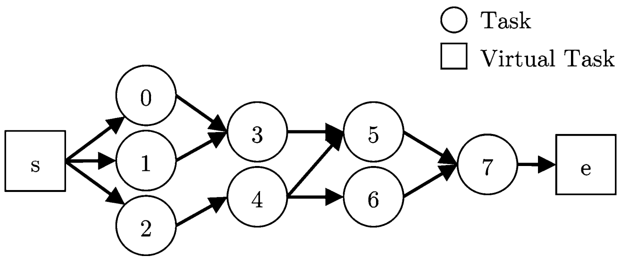

Fig. 1 Directed acyclic graph (DAG) [27]

$$\begin{aligned} &minimize\,(f = T_{Makespan} + \sum_{i=1}^{n}\sum_{j=1}^{n} E_{i,j}) \\ &subject\,to\,(T_{Makespan} \leq U(s)\,,\ Cost \leq B) \end{aligned} \tag{1}$$

in which f and $T_{Makespan}$ show the objective function and Makespan of each task. Moreover, n and m are the number of tasks and the number of VMs, respectively. $E_{i,j}$ means the cost of executing task i on VM_j. It is important that $T_{Makespan}$ should be less than the number of overdue task (U) by considering the main constraint of $Cost = \sum_{j=1}^{m} C_{i,j}$ less than the total allocated budget(B).

Scheduling problem is studied traditionally through different methods including Greedy (First-Fit), round robin algorithm used in Eucalyptus cloud system [20], Advanced reservation and preemption scheduling [32], earliest due date first (EDD) [8], critical path method (CPM) [17], project evaluation and review technique (PRET) [13], dynamic programming [25], and branch-and-bound [26, 29, 30].

However, these algorithms do not take into account the optimal resource allocation and load balancing aiming in the minimal response time. Besides, first-in-first-out (FIFO) is used as the default scheduling algorithm in Hadoop without considering fairness [16]. However, it causes long response time in the case of being the number of tasks much more than the capacity of the computation resources. Furthermore, improved Hadoop Fair scheduler (HFS) with delay scheduling [31] applied by Facebook and capacity scheduler [28] used by Yahoo are the main scheduling algorithms. However, these two algorithms emphasize finding response time for each job of the workflow without considering the minimal response time.

Min-Min and Max–Min algorithms deal with ETC matrix with $M \times N$ in which M and N indicate the number of tasks and the number of resources, respectively. Although these algorithms could be implemented easily due to considering few rules to allocate resources to tasks, they suffer from a lack of finding the optimal solution in a reasonable time while considering multi-criteria in the complex situation. Therefore, meta-heuristic and hybrid heuristic algorithms could be more efficient to solve the scheduling problem in a reasonable time [27, 28].

3 Hyper-heuristic scheduling algorithms on cloud computing

Scheduling on the cloud has been widely studied over the years and is defined as an NP problem by considering multi-purpose. Therefore, it is difficult to find an optimal solution within polynomial time and scheduling algorithm attempts to find a near-optimal solution. Heuristic algorithms due to using tactical estimates can provide better solutions than rule-based algorithms for complex scheduling problems on cloud [11, 28].

Ant colony optimization [7] was applied to provide a better scheduling solution than other scheduling algorithms with aiming to minimize the execution cost of a single workflow and to balance loads on different resources [7]. However, the execution time of workflow is not considered as one of the objectives in scheduling algorithms as well as elasticity of the cloud with different numbers and types of virtual machines is ignored. Moreover, PSO is used to solve the workflow scheduling problem. However, it takes much more computation time than the others Or Simulated Annealing algorithm [15] is less computation time but face the problem of getting stuck at a local optimum.

In the case of uncertainty, such as working environment variation, machine breakdown, unstable workers, etc., the job time is not a fixed number. To minimize the maximum total completion times, a branch and bound algorithm is applied to find the optimal solution in [29]. Moreover, a simulated annealing hyper-heuristic (SA-HH) incorporating the proposed seven low-level heuristics has been developed to solve the two-scenario-based dependent processing times on a single machine. To solve the flowshop scheduling problem and minimize the maximum Makespan, eight polynomial heuristics has been developed using Johnson's rule in [30]. Furthermore, four cloud theory-based simulated annealing (CSA) hyper-heuristic algorithms are proposed incorporating seven low level heuristics to solve a robust two-stage assembly flowshop problem with scenario-dependent processing times.

Furthermore, a Combination of several heuristic algorithms is applied to provide a strong scheduling algorithm which is known as hybrid heuristic algorithms [28]. Moreover, the Hyper-heuristic algorithm leverages the scheduling algorithm using the complementary advantages of heuristic algorithms to find the optimal solution [27].

Figure 2 demonstrates the operations of the heuristic algorithm, hybrid heuristic algorithm, the hyper-heuristic algorithm as represented in [27]. The heuristic algorithm performs three phases including transition (T), evaluation (E), and determination (D) in each iteration until reaching the maximum number of iterations. The hybrid heuristic algorithm selects heuristic algorithm (h_i) for performing each phase of transition (T), evaluation (E), and determination (D) in each iteration of convergence process with the maximum number of t causing to increase long computation time. To reduce the computation time, Hyper-heuristic algorithm is provided to choose one heuristic algorithm (h_i) from candidate heuristic algorithm set $\{H_1, H_2, \ldots, H_n\}$ by the low-level heuristic (LLH) selection operator in each iteration including three phases [27]. Selected (h_i) will be changed due to sticking with a solution for a certain number of iteration or falling into a local optimum. To do so, the acceptance operator is chosen to define the Selected (h_i) will be continued for the next iteration. Choosing (h_i) by LLH operator and determining to be continued for the next iteration by acceptance operator are two major components of a hyper-heuristic algorithm. Selecting one heuristic algorithm in each iteration maintains a high search diversity resulting in increasing the chance of finding a better solution as well as degrading the computation time.

The important metric of the performance scheduling algorithms time complexity. However time complexity of the heuristic algorithm is expressed by $O(LNM^2)$ where L and N and M defines the number of iterations, the population size and the number of sub-solutions, $O(LNM^2)$ is considered as the time complexity of the hybrid heuristic algorithm in which assuming that the number of states $M \times N$ (M represents the number of virtual machines and N is the number of tasks) of the hybrid heuristic algorithm equals M of the heuristic algorithm and assuming that the time complexity of all the other operators of heuristic algorithms are bounded from above

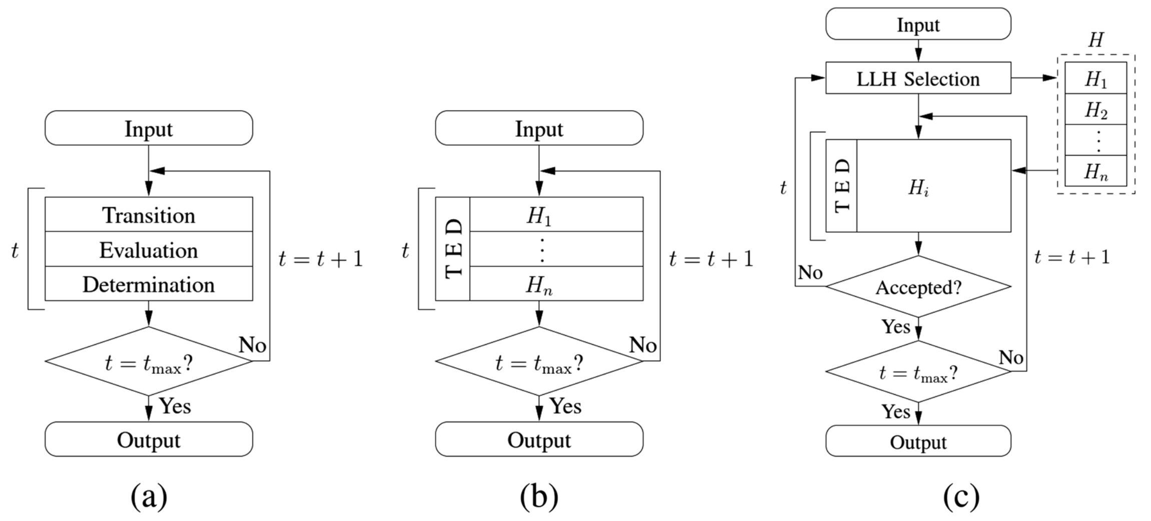

Fig. 2 Outline of **a** heuristics, **b** hybrid-heuristics, and **c** hyper-heuristics [27]

by NM^2. The time complexity of the hyper-heuristic algorithm is also $O(LNM^2)$ if all the operators (or heuristic algorithms) used are bounded from above by NM^2 [27].

For a closer look to computation time on heuristic, hybrid heuristic and hyper-heuristic algorithms in scheduling problem, If assume that computation time in each iteration of heuristic algorithms is $O(X)$ (X is a hypothetical value) and T is the number of iterations, computation time in total repetitions for the heuristic algorithm is equal to $O(T.X)$, and for hybrid heuristic algorithms, computation time in each iteration is $O(X) = O(X)_1 + O(X)_2 + \ldots + O(X)_i + \ldots + O(X)_n$ ($O(X)$ denotes the computation time of the heuristic algorithm H_i and n denotes the number of heuristic algorithms which is combined) and total repetitions is $O(T.X) = O(T.X)_1 + O(T.X)_2 + \ldots + O(T.X)_i + \ldots + O(T.X)_n$. Such as for the hyper-heuristic algorithms in each iteration, one and only one heuristic algorithm runs, therefore, computation time in total repetitions is equal to $O(T.X) = O(t_1.X)_1 + O(t_2.X)_2 + \ldots + O(t_i.X)_i + \ldots + O(t_n.X)_n$ (t_i denotes the number of repetitions in heuristic is selected by the operator LLH). Because the value of t_i is less than or equal T, therefore, the computation time of hyper-heuristic algorithms is usually less than the hybrid heuristic algorithms. Since working hyper-heuristic algorithms is finding a suitable heuristic from their own the set of candidate heuristics and its implementation, so that, so that hyper-heuristic algorithms have an optimal solution, they must limit the number of iteration Selected heuristic at the level of a single heuristic algorithm. Namely, value repetition of the hyper-heuristic algorithm must be several times a single heuristic algorithm, because after selecting a heuristic by LLH, usually previous heuristic before replacing with selected heuristic, is repeated several times, and that means the number of repetitions for the selected heuristic equal with $T = T - t_i - 1$ ($t_i - 1$ denotes the number of repetitions the previous heuristic). These make the computation time hyper heuristic is more than the single heuristic algorithm. In continue, to reduce the computation time hyper-heuristic algorithms to level computation time single heuristic algorithm proposed a new method.

In the hyper-heuristic algorithm, LLH is performed randomly without considering the characteristics of input workflow leading to performance degradation. In this paper, we propose an efficient method based on a decision tree to find the suitable heuristic algorithm from a pool of candidate heuristic algorithms depending on the characteristics of input workflow resulting in a reduction of computation time. We are aiming at (a) minimizing Makespan, and (b) achieving maximum accuracy in comparison with hyper-heuristic algorithm due to selecting the optimal heuristic algorithm from the candidate set based on the specifications of workflow through utilizing decision tree.

4 Hyper-heuristic selector algorithm

In this section, we propose an efficient method to enhance the hyper-heuristic scheduling algorithm to reduce Makespan and to improve accuracy in the resource allocation mechanism to tasks of the application.

The aim of hyper-heuristic selector algorithms is to find an appropriate heuristic algorithm from their own candidate heuristic set for an incoming workflow dependent on the specifications of input workflow. To do so, different heuristic algorithms should be executed on the various input workflows to identify which a heuristic algorithm is a suitable option. Besides, applications running on the cloud could be classified into two main groups including the same specifications and different particular characteristics of the workflow. Since Applications running on the cloud could be considered with the same structure of workflow, hyper-heuristic selector algorithm attempts to select a suitable heuristic algorithm for each workflow using classification and clustering techniques based on prior knowledge of the performance of all candidate heuristics set in a hyper-heuristic algorithm on entered workflows to a cloud.

Algorithm 1 explains the hyper-heuristic selector algorithm in detail. The hyper-heuristic selector algorithm attempts to find the optimal heuristic algorithm from its own candidate set through extracting the best heuristic algorithm from the database of different input workflows using the decision tree approach. As far as the proposed algorithm described here, the candidate pool of scheduling algorithm includes a Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). Since heuristic algorithms are based on probabilities, it is more difficult to define which heuristic algorithm performs the best in the set of the candidate pool. Although the performance of some heuristic algorithms such as ant colony optimization could be much better than simulated annealing in the topic of scheduling, it is not considered for scheduling some workflows, it could not be performed the best for some workflows. Therefore, the Enhanced hyper-heuristic algorithm method extracts rules to select the appropriate algorithm by LLH operator using classification techniques such as Decision Tree on a dataset including samples from incoming workflows to a cloud. Each workflow is labeled with the appropriate heuristic algorithm in clustering.

Algorithm 1 Hyper Heuristic Selector Algorithm.

```
1: Inputs: The scheduling problem consisting of workflow and numbers of cloud resources,
database of different input workflows;
2: Outputs: the best heuristic algorithm
3: Setup the parameters;
4: Initialize the population of solutions as the candidate set {$h_1, h_2, ..., h_n$};
5: select a heuristic algorithm hi from the candidate set based on the type of input workflow
using LLH operator
6: while the termination condition is not met do
7: Update the probability of Solutions H by using the selected algorithm
8: end while
9: return the best solution
```

The structure of the Enhanced hyper-heuristic algorithm is shown in Fig. 3, in which its acceptance operator is removed compared with the hyper-heuristic algorithm and LLH operator does the acceptance action, too. the LLH operator selects the appropriate heuristic algorithm from the candidate set based on the decision tree carried on the database of different workflows labeled with the suitable heuristic

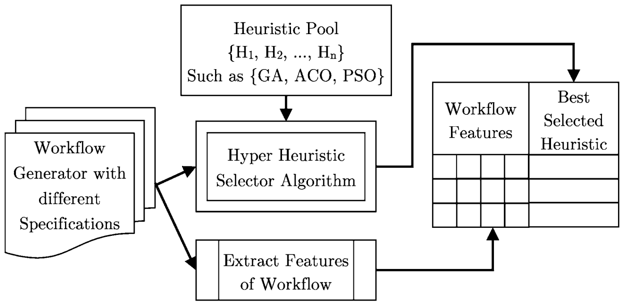
Heuristic Pool
{H1, H2, ..., Hn}
Such as {GA, ACO, PSO}
Workflow Generator with different Specifications
Hyper Heuristic Selector Algorithm
Extract Features of Workflow
Workflow Features
Best Selected Heuristic

Fig. 3 The structure of Hyper Heuristic Selector Algorithm

algorithm. Moreover, the acceptance operator is ignored to determine if the selected heuristic algorithm should be continued for the next iteration. Therefore, the computation time of the algorithm is reduced due to limiting to only the selection of heuristic by LLH operator.

To create a dataset to find a pattern and rules, different workflows entered into the cloud are used. A workflow is identified by different features including the total number of tasks, the number of children tasks and parent tasks, the number of edges, mean duration of tasks, etc., making a distinction among workflows. Moreover, the created datasets consist of cloud features defined by the number of virtual machines in the cloud which be accounted for workflows.

As mentioned, the proposed algorithm using the classification and clustering removes the searching process to find the best heuristic algorithm. The enhanced hyper-heuristic algorithm selects the appropriate heuristic algorithm based on the probability of different types of heuristic algorithms applied to a different type of workflows and characteristics of resources of cloud allocated to different kinds of the workflow. It causes that the computation time of the enhanced hyper-scheduling algorithm is equal to the computation time of selected algorithm $\mathrm{O}(T.X) = \mathrm{O}(T.X)_i$ which H_i is the best-selected heuristic algorithm.

5 Simulation and experimental results

To make an example, Workflowsim simulator [5] and RapidMiner software [23] as the main tools are used to simulate the cloud computing environment and data mining.

Workflowsim settings depicted in Table 1 are identified by the specifications of cloud data centers and their virtual machines. Cloudsim is used to create datacenters with its virtual machines. Datacenter characteristics are identified by the numbers of data centers and employed hops in each datacenter as well as applied suitable algorithms (tag class for scheduling). Besides, the specifications of cloud virtual machines consist of the number of virtual machines, average and standard deviation processing speed of virtual machines as well as bandwidth among VMS. As shown in Table 1, it is assumed that the number of VMs is considered to be 3 or 4 with

Table 1 Workflowsim settings

Virtual Machine	Number of Virtual Machines	3–4
	MIPS	100–600
	RAM	512
	Bandwidth	150
	Cloudlet Scheduler	Space-shared
	Number of cores	1
Datacenter	Number of Datacenters	1
	Number of Hosts	2
	VM Scheduler	Time-Shared

their speed varying between 100 to 600MIPS. Also, in this example, for the simplicity of implementation, it is assumed a shared memory to ignore the communication cost among virtual machines.

In addition to defining cloud data centers and their virtual machines, the characteristics of workflows should be identified. Furthermore, two types of workflows including SIPHT with 30 nodes and Montage with 25 nodes are employed to create a dataset. Dataset features presented in Table 2 are implied by volume, type and status (input, output). Moreover, to create a dataset, each sample of the dataset should be labeled. To do so, each type of heuristic algorithm in the heuristic candidate set is carried out 10 times on each sample. The algorithm with the lowest average Makespan is considered as the label of each sample. Besides, it should be noted that the name of workflows is used instead of workflow features. However, it is better to use different features of workflow for a large number of workflows. Besides, each simulation is carried out 30 runs on each workflow.

In this example, it is assumed that the candidate heuristic set is composed of H = {Genetic (Ga) [12], Ant Colony Optimization (ACO) [7], Particle Swarm Optimization (PSO)[14]}, which have been implemented for simulation of Makespan fitness function. Parameter settings of each heuristic algorithm are presented in Table 3. All parameters have been set according to the based paper [27].

The Makespan results of applying heuristic algorithms on SIPHT and Montage datasets are shown in Fig. 4. the results show, the best algorithm for the SIPHT dataset is the GA and the best choice for the Montage dataset is PSO.

After defining the datasets, a decision tree is applied to each classification of samples to extract rules for workflows. The rules of the decision tree on 100 workflow samples in RapidMiner are as follows:

```
Type = SIPHT
    SD-MIPS ≥ 34.200
        Avg-MIPS ≥ 329: GA
        Avg-MIPS ≤ 329: PSO
    SD-MIPS ≤ 34.200
        GA
Type = Montage
    SD-MIPS ≥ 81.500: GA
    SD-MIPS ≤ 81.500: ACO
```

According to the extracted rules from the decision tree, the operator LLH selects the appropriate heuristic algorithm from candidate heuristic set for new arrival workflows to the cloud. For instance, if Montage workflow type is implemented on 4 virtual machines with a standard deviation of 100 and an average speed of 500, the best algorithm for scheduling will be a genetic algorithm.

Figure 5 shows a diagram of the mentioned rules to make a decision tree. According to Fig. 5, depending on which condition is met, the suitable heuristic algorithm is chosen from the candidate heuristic set. In the case of workflow with type A, when std-MIPS > 34.200 and avg-MIPS > 329 occurs, the appropriate heuristic algorithm from candidate heuristic set is GA through following decision tree. However, GA algorithm is considered as the best algorithm in the mentioned conditions, its accuracy could not be reached to 100% referred to the red color in the most left leaf of the decision tree in Fig. 5. Therefore, to increase accuracy in the selection of heuristic algorithms from a set of candidate heuristic algorithms, it is necessary to consider effective parameters including the number of a sample dataset, the number of features, proper selection of features workflows, etc.

5.1 Accuracy vs. Recall

This example deals with two cases: one dataset with 100 samples and the other test dataset with 20 samples. More specifically, Fig. 6 shows the comparison of the accuracy of decision tree rules for different heuristic algorithms of candidate heuristic set such as GA, ACO, PSO. In this example, the overall accuracy for decision tree rules from a dataset with 100 samples and a test dataset with 20 samples is estimated to be 87%. The result implies if the ACO algorithm is chosen as the heuristic algorithm, the accuracy of scheduling could be improved by 85.7% through the rules extracted

Table 2 Examples of dataset

Type	Con-VM	Avg-MIPS	Sd-MIPS	Scheduling algorithm
SIPHT	4	340	179	GA
Montage	3	203	63	PSO

Table 3 Parameter settings of heuristic algorithms

Scheduling Algorithm	Properties
GA	mutation probability $m = 0.95$ iteration algorithm t = 100
ACO	pheromone updating $\rho_{global} = 0.1$ $\rho_{local} = 0.9$ iteration algorithm t = 50 choosing probability $q_0 = 0.85$ related influence weights $\alpha = 1$, $\beta = 0.95$
PSO	inertia weight $\omega = 0.8$ acceleration coefficient $c_1 = c_2 = 1.0$ iteration algorithm t = 100

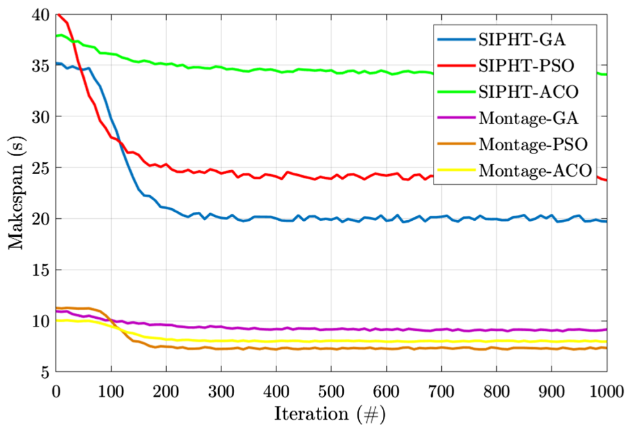

Fig. 4 Makespan results of applying GA, PSO, ACO on SIPHT and Montage dataset

from the decision tree. Whereas the accuracy for ACO is 85.71% and the least accuracy is for GA with 72.73%. The most important reason for GA's low accuracy is the higher number of samples related to GA class. This is exactly why the accuracy of the PSO algorithm has reached a maximum of 100% due to the smaller number of samples.

Figure 6 also presents the class recall chart of a decision tree concerning different types of heuristic algorithms. For example, the percentage of those from the test dataset which decision tree is correctly attributed to the label or class PSO algorithm is 50%, whereas the PSO accuracy was 100%. According to the decision tree, the leftmost leaf is labeled with GA, but it's below bar shows that a significant number of samples belong to the PSO class. This conflict decrease the PSO recall. On the other hand, the greater number of samples in the GA class has led to a sharp increase in the amount of its recall.

5.2 Hyper-heuristics scheduling algorithm (HHSA) vs. decision tree hyper-heuristics selector (DT-HHS)

To compare the selection time between the Low-Level Hyper-heuristics selector represented in [27] and the proposed Decision Tree-based Hyper-Heuristics Selector, the 100 workflow samples from SIPHT and Montage workflow types have been generated with Workflowsim simulator. All workflows were assigned to both algorithms, and the Hyper-heuristics selection time and Makespan were measured in both systems. The Makespan of HHSA, DT-HHS, their difference, HHSA Selection Time, and HHS Selection Time is shown in Fig. 7 for Montage and SIPHT Workflows, respectively.

The selection time in the proposed DT-HHS includes three steps: 1. Workflow feature extraction, 2. Run the Decision Tree model to find the best heuristic, and 3. Run the selected Heuristic. The selection time is usually fixed because exactly one heuristic is chosen. On the other side, the HHSA does not include the first two steps of feature extraction and running model. In some cases, it succeeds on its first choice. It means that the first selected heuristic is the most suitable one. But in many cases, HHSA should try to test more than one heuristic to find a suitable one. Because the first two steps of the DT-HHS are very fast and do not take long, they can be ignored. The result demonstrates the lower time complexity of the DT-HHS as mentioned. The average selection time for DT-HHS is 1.9982 s against 3.0695 s for HHSA. But it should be noted that because HHSA tests all heuristics, it will always choose the best heuristic and therefore have a better Makespan. However, DT-HHS with an accuracy of 87%, will make the wrong decision in some cases. As a result, if it does not choose the right heuristic, the Makespan will increase. This phenomenon can be seen in the results in the form of graph peaks. According to the results, this difference is insignificant and can be neglected due to the faster selection time. In most cases, the same Makespan is seen for DT-HHS and HHSA, and the slight differences are due to the randomness of the heuristics. Finally, It should be mentioned that there is a trade-off between Selection Time and Makespan. The

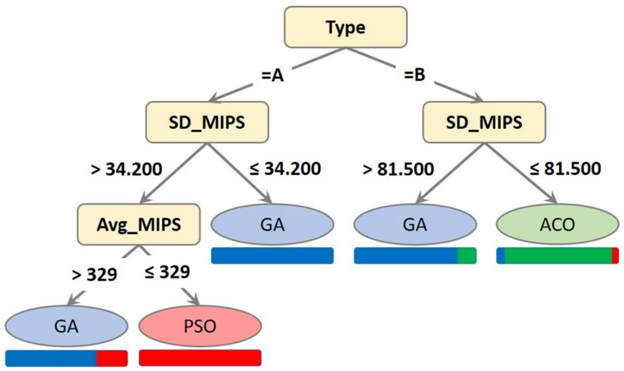

Fig. 5 Decision tree

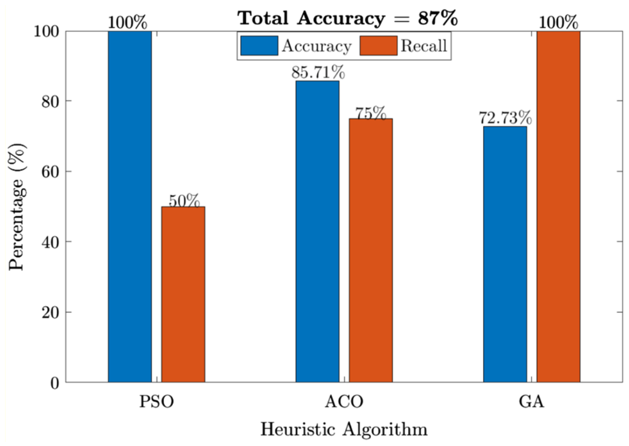

Fig. 6 The accuracy and Recall of heuristic algorithms

HHSA shows better Makespan in contrast with the better Selection Time of the proposed Decision Tree-Based Hyper-Heuristic Selector.

6 Conclusion

Heuristic algorithms are the perfect solution for complex scheduling problems. Each of these algorithms has its own pros and cons, and this has caused researchers, with the aim of the use of their complementary pros, to combine them to create new algorithms. One of the composition methods is using of the hyper-heuristic heuristic algorithms, that in this paper, by using the rules of classification dataset of entered workflows into a cloud and virtual machines, a new hyper-heuristic algorithm was presented, that its computation time is reduced by removing the acceptance operator to level computation time single heuristic algorithm, and increases the power of decision for the assignment virtual machines to users with specific limitations according to the type of workflows. Of research works, which is recommended for better performance of this algorithm, are finding appropriate methods of classification and suitable candidate.

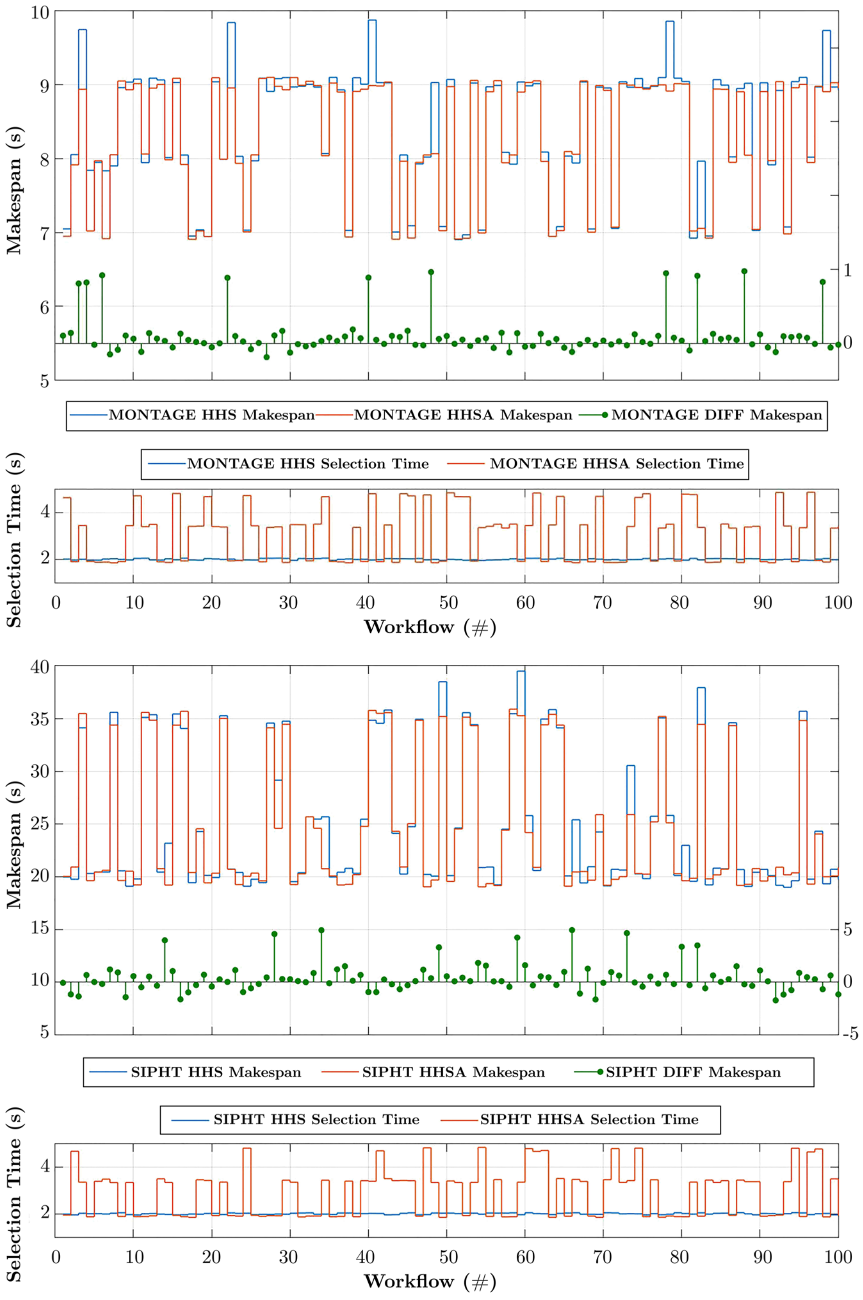

Fig. 7 Selection Time and Makespan of HHSA vs. DT-HHS

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Allahverdi, A., Ng, C., Cheng, T.E., Kovalyov, M.Y.: A survey of scheduling problems with setup times or costs. Eur. J. Oper. Res. **187**(3), 985–1032 (2008)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
3. Bilal, K., Manzano, M., Khan, S.U., Calle, E., Li, K., Zomaya, A.Y.: On the characterization of the structural robustness of data center networks. IEEE Trans. Cloud Comput. **1**(1), 1–1 (2013)
4. Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J.: Scheduling computer and manufacturing processes. Springer science & Business media, Berlin (2013)
5. Chen, W., Deelman, E.: Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In: 2012 IEEE 8th International Conference on E-Science, IEEE, pp. 1–8. (2012)
6. Chr´etienne, P., Coffman, E., Lenstra, J., Liu, Z.: Scheduling Theory and Its Applications. Wiley (1995). URL https://books.google.com/books?id=pvVTAAAAMAAJ
7. Dorigo, M., Gambardella, L.M.: A cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
8. Elsayed, K.M., Khattab, A.K.: Channel-aware earliest deadline due fair scheduling for wireless multimedia networks. Wirel. Pers. Commun. **38**(2), 233–252 (2006)
9. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: 2008 grid computing environments workshop, IEEE, pp. 1–10. (2008)
10. Garey, M.R., Johnson, D.S.: Computers and intractability, vol. 174. Freeman San Francisco, San Francisco (1979)
11. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Math. Oper. Res. **1**(2), 117–129 (1976)
12. Holland, J.H., et al.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT press, Cambridge (1992)
13. Ingalls, R.G., Morrice, D.: Pert scheduling with resources using qualitative simulation graphs. In: 2000 Winter Simulation Conference Proceedings (Cat. No. 00CH37165), vol. 1, pp. 362–370. IEEE (2000)
14. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95—International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)
15. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. science **220**(4598), 671–680 (1983)
16. Kittusamy, K., Balasubramanie, P.: Ant algorithm for grid scheduling powered by local search. Int. J. Open Prob. Comput. Sci. Math. **3**, 222–240 (2008)
17. Kohler, W.H.: A preliminary evaluation of the critical path method for scheduling tasks on multiprocessor systems. IEEE Trans. Comput. **100**(12), 1235–1238 (1975)
18. Lei, X., Liao, X., Huang, T., Li, H., Hu, C.: Outsourcing large matrix inversion computation to a public cloud. IEEE Trans. Cloud Comput. **1**(1), 1–1 (2013)
19. Leung, J.Y.: Handbook of scheduling: algorithms, models, and performance analysis. CRC Press, Boca Raton (2004)
20. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE pp. 124–131 (2009)
21. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Springer International Publishing (2018). URL https://books.google.com/books?id=Z6SPtQEACAAJ
22. Rahman, M., Li, X., Palit, H.: Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment. In: 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, IEEE, pp. 966–974. (2011)
23. Ramamohan, Y., Vasantharao, K., Chakravarti, C.K., Ratnam, A., et al.: A study of data mining tools in knowledge discovery process. Int. J. Soft Comput. Eng. **2**(3), 2231–2307 (2012)

24. Saovapakhiran, B., Michailidis, G., Devetsikiotis, M.: Aggregated-dag scheduling for job flow maximization in heterogeneous cloud computing. In: 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011, IEEE, pp. 1–6. (2011)
25. Schuetz, H.J., Kolisch, R.: Approximate dynamic programming for capacity allocation in the service industry. Eur. J. Oper. Res. **218**(1), 239–250 (2012)
26. Shi, D., Chen, T.: Optimal periodic scheduling of sensor networks: a branch and bound approach. Syst. Control Lett. **62**(9), 732–738 (2013)
27. Tsai, C.W., Huang, W.C., Chiang, M.H., Chiang, M.C., Yang, C.S.: A hyper-heuristic scheduling algorithm for cloud. IEEE Trans. Cloud Comput. **2**(2), 236–250 (2014)
28. Tsai, C.W., Rodrigues, J.J.: Metaheuristic scheduling for cloud: a survey. IEEE Syst. J. **8**(1), 279–291 (2013)
29. Wu, C.C., Bai, D., Chen, J.H., Lin, W.C., Xing, L., Lin, J.C., Cheng, S.R.: Several variants of simulated annealing hyper-heuristic for a single-machine scheduling with two-scenario-based dependent processing times. Swarm Evol. Comput. **60**, 100765 (2020)
30. Wu, C.C., Gupta, J.N., Cheng, S.R., Lin, B.M., Yip, S.H., Lin, W.C.: Robust scheduling for a two-stage assembly shop with scenario-dependent processing times. Int. J. Prod. Res. pp. 1–16 (2020)
31. Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., Stoica, I.: Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In: Proceedings of the 5th European Conference on Computer systems, pp. 265–278 (2010)
32. Zhong, H., Tao, K., Zhang, X.: An approach to optimized resource scheduling algorithm for open-source cloud systems. In: 2010 Fifth Annual ChinaGrid Conference, IEEE, pp. 124– 129. (2010)