

# An efficient cloudlet scheduling via bin packing in cloud computing

Amine Chraibi, Said Ben Alla, Abdellah Ezzati

Faculty of Science and Technology, Laboratory of Emerging Technologies Monitoring, Hassan First University of Settat, Settat, Morocco

---

## Article Info

### Article history:

Received May 10, 2021

Revised Jan 14, 2022

Accepted Jan 31, 2022

---

### Keywords:

Bin packing problem

Cloud computing

CloudSim

Particle swarm optimisation algorithm

Task scheduling

---

## ABSTRACT

In this ever-developing technological world, one way to manage and deliver services is through cloud computing, a massive web of heterogenous autonomous systems that comprise adaptable computational design. Cloud computing can be improved through task scheduling, albeit it being the most challenging aspect to be improved. Better task scheduling can improve response time, reduce power consumption and processing time, enhance makespan and throughput, and increase profit by reducing operating costs and raising the system reliability. This study aims to improve job scheduling by transferring the job scheduling problem into a bin packing problem. Three modified implementations of bin packing algorithms were proposed to be used for task scheduling (MBPTS) based on the minimisation of makespan. The results, which were based on the open-source simulator CloudSim, demonstrated that the proposed MBPTS was adequate to optimise balance results, reduce waiting time and makespan, and improve the utilisation of the resource in comparison to the current scheduling algorithms such as the particle swarm optimisation (PSO) and first come first serve (FCFS).

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



---

## Corresponding Author:

Said Ben Alla

Department of Mathematics and Computer Science, Hassan First University of Settat

577 Casablanca Road, Settat, Morocco

Email: saidb\_05@hotmail.com

---

## 1. INTRODUCTION

Cloud computing provides on-demand services, including networks, servers, storage, and applications through its massive and effective computing paradigm. The National Institute of Standards and Technology (NIST) defines it as a developing technology that frequently offers accessible and on-demand network access to shared computing resources [1]. The typical models of the cloud are: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) [1].

Apart from that, task scheduling, which has gained traction nowadays, introduces the option of choosing the resources distribution between various tasks. It should be noted that each workflow or tasks may have scalable scheduling on multiple virtual machines (VMs). Regarding task scheduling, its nondeterministic polynomial time (NP) nature may cause issues that stemmed from the resources' unstable characteristics and dynamic nature [2]. In the process, the task scheduler accepts the queued tasks from the users and assigns the tasks to available resources based on the task resources parameters [2]. The research problem is to improve task scheduling in cloud computing by reducing the execution time of queuing tasks and enhancing the use of resources.

There have been many studies recently on ways to improve task scheduling in cloud computing. The studies rely on algorithms that demonstrate their efficiency in task scheduling. For reducing energy consumption and making a whole system live longer, Kumar and Alam [3] proposed a novel scheduling algorithm based on dynamic voltage scaling (DVS) and earliest deadline first (EDF) algorithm. Ouham and Yadi [4] proposed a new technique to improve the data allocation system in virtual machine (VM) for cloud computing based on a modified grey wolf optimization algorithm. Aziz and Ninggal [5] propose a failure-aware workflow method for scheduling parallel applications to improve the reliability and makespan metrics on homogeneous systems. To improve the makespan, Aziz *et al.* [6] developed a failure-aware workflow scheduling algorithm to reprocess the failed job and assign them to the unused resources. It is also observed that the total cost can be reduced through an energy consumption model that is composed of the processor's execution and transmission cost [7]. Furthermore, the improvement of the load deviation, the resource utilisation (RU) and the makespan can be achieved through the proposed algorithm in the paper [8] denoted hybrid load balance based on genetic algorithm (HLBGA). Similarly, resource utilisation, load balancing, and excellent performance can also be achieved by implementing a novel architecture that is based on the particle swarm optimisation (PSO) and the dynamic dispatch queues algorithms [2]. Moreover, Harun *et al.* [9] introduced a genetic algorithm GA-based task scheduler algorithm for a mobile robot in the ground to find the optimum global travel itinerary for picking and delivering products at different locations. The execution time of task scheduling and the throughput of cloud computing can also be improved through a novel dynamic task scheduling algorithm using GA by considering the scalability of the cloud [10]. In addition, the total execution time can also be reduced by using the algorithm introduced by Gabi *et al.* [11] that is based on the orthogonal Taguchi-based cat swarm algorithm. Apart from that, the hybrids of simulated annealing (SA) with PSO and fuzzy logic with PSO have been demonstrated to impact makespan, waiting time, and other metrics positively [12]. Also, in order to reduce the waiting time of tasks in a queue's set, another smart scheduler is proposed by Abdalkafor and Alheeti in the paper [13]. Additionally, to improve the latency, response time, and amount of data used in a fog node Alsmadi *et al.* [14] proposed a RR based scheduler. To minimise the overall execution time of a set of tasks of a directed acyclic graph (DAG), Edward and Elcock [15] proposed an algorithm based on the ant colony optimisation (ACO) algorithm denoted ranking-ant colony system (rACS). Krishnadoss and Jacob [16] proposed a hybrid algorithm for task scheduling based on oppositional-based learning and Cuckoo search algorithm to assign users tasks and minimise the cost and makespan of the system. Khorsand and Ramezanpour [17] proposed an improved task scheduler based on the best-worst methods (BWM) and the technique for order preference by similarity to ideal solution (TOPSIS) to optimise metrics like energy consumption, makespan. In [18] developed an efficient task scheduling method to enhance resource efficiency and fault tolerance utilising a dynamic load-based distributed queue for dependent jobs. In [19] developed an ACO method to select the best virtual machine for executing a cloudlet to reduce energy consumption and execution time. We also proposed two tasks scheduling works [20] and [21]; the first accelerated the PSO task scheduling algorithm, and the second improved the makespan and other performance metrics using deep q-learning.

As solutions, this study proposes and compares novel alternatives with PSO. The proposed alternatives were acquired by converting the task scheduling problem into a bin packing problem. In addition, the bin packing algorithm was utilised by comparing measures of task scheduling such as waiting time, makespan, and resource utilisation. In order to improve the reliability of the proposed method, we compare the results of our proposed algorithms with the PSO, the most popular algorithm in the scheduling field. We also used the first come first serve (FCFS), which is the default scheduling algorithm used in CloudSim. Section 2 of this paper describes studies related to the current research, while section 3 describes the proposed algorithms, in which its experimental setup and simulation results are discussed in section 4, section 5 concludes the paper.

## 2. PROPOSED TASK SCHEDULING ALGORITHMS

### 2.1. Task scheduling problem

Task scheduling is an essential approach in cloud computing to resolve many issues, especially the overlay of cloud provider on the users' requirements such as maximum profit and quality of service (QoS) [22]. The cloud provider attempts to minimise waiting time and effectively utilise the VMs while simultaneously minimising the makespan. Figure 1 shows that a significant set of distinct tasks with varying parameters are raised by various users for the cloud provider to manage, which are later assigned to the available VMs.

In this regard, various optimisation algorithms are used to fully utilise VMs and find a better approach

to reduce the overall execution time, which is our primary objective. To do so, we depend on (1) in calculating the makespan and (2) in calculating the execution time in a single VM  $j$  [23].

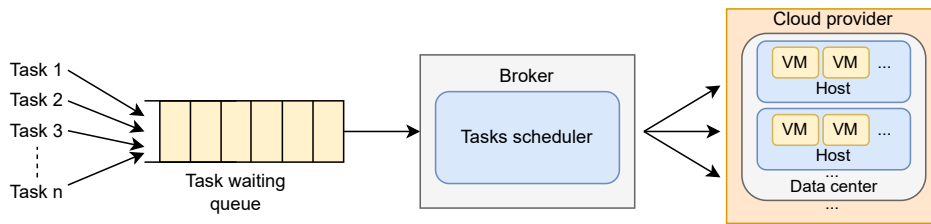


Figure 1. Cloud scheduling architecture

$$Makespan = \max(Ex_{VM_1}, Ex_{VM_2}, \dots, Ex_{VM_j}, \dots, Ex_{VM_m}) \tag{1}$$

$$Ex_{VM_j} = \sum_{i=0}^n Ex_j(Cloudlet_i) \tag{2}$$

where,  $VM_j$  is the VM  $j$ ,  $Cloudlet_i$  is the processing power of task  $i$  in million instructions per second (MIPS) and  $Ex_j(Cloudlet_i)$  is the execution time of cloudlet  $i$  on  $VM_j$ ,  $Makespan$  is the overall execution time. Figure 2 shows an example of the process using FCFS algorithm where the number of VMs is 2, and the number of tasks is 7.

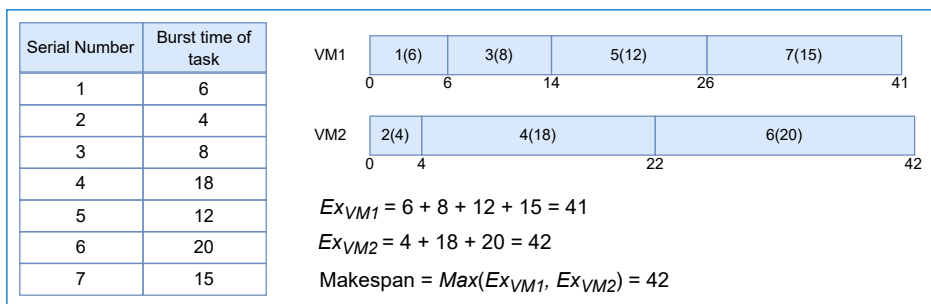


Figure 2. Makespan calculation example

### 2.2. Bin packing

Bin packing problem is an infamous issue in combinatorial optimisation [24]-[26]. It pertains to the given  $n$  objects (items) of various values and containers (bins). Each has a max capacity  $C_{\max}$ . The primary purpose is to allocate each object to a container in ways that will minimise the total number of used containers. It can be considered that the objects have smaller values than container capacity. The following (3) is used [27]:

$$\begin{aligned}
 & \min \sum_{i=1}^n y_i \\
 & \text{subject to } \sum_{j=1}^n w_j x_{ij} \leq C_{\max} \times y_i \quad i = 1, \dots, n \\
 & \sum_{i=1}^n x_{ij} = 1 \quad j = 1, \dots, n \\
 & y_i \in \{0, 1\} \quad i = 1, \dots, n \\
 & x_{ij} \in \{0, 1\} \quad i = 1, \dots, n \quad j = 1, \dots, n
 \end{aligned} \tag{3}$$

where,  $y_i = 1$  shows that container  $i$  is operated ( $y_i = 0$  otherwise), and  $x_{ij}$  shows that object  $j$  should be packaged in container  $i$  and  $w_j$  the weight of object  $j$ . In addition, the constraint  $\sum_{i=1}^n x_{ij} = 1$  assures that

each object  $j$  is packed just one time, while inequalities  $\sum_{j=1}^n w_j x_{ij} \leq C_{\max} \times y_i$  ensure that the capacity  $C_{\max}$  constraint is respected for all containers that are used [27]. Figure 3 illustrates the bin packing problem. We have seven initial objects of different sizes that we want to optimally place on the four given containers to use only a few of them.

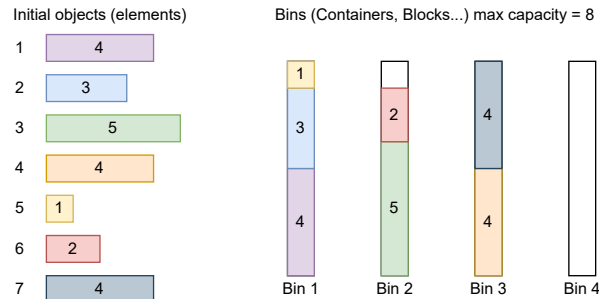


Figure 3. Illustration of the bin packing problem

### 2.3. Proposed algorithms

This study focuses on a novel method for task scheduling based on bin packing problem algorithms such as first-fit, next-fit, and best-fit algorithms [28]. It aims to improve efficiency through makespan optimisation, reducing waiting time, improving load balancing, and increasing resource utilisation. The approaches proposed in this study intent on reformulating the task scheduling problem to construct a bin packing problem by considering the VMs as bins with a max capacity  $C_{\max}$ , in addition to considering the tasks as items to be placed into those bins based on their length of tasks, which is also referred to as the weight of the item. The proposed approaches also aim to minimise makespan – the maximum capacity to be used in the bin packing problem – instead of minimising the number of bins. The maximum capacity of the bins is calculated using the proposed (4) which we consider as the perfect makespan to approach. Where,  $T_i$  represents the average execution times of task  $i$  on each available VM,  $m$  denotes the number of available VMs, while  $n$  is the total number of tasks.

$$C_{max} = \frac{\sum_{i=1}^n T_i}{m} \quad (4)$$

After defining the bin packing parameters, the bin packing algorithm is implemented to solve the problem. Figure 4 describes the workflow of the whole scheduling process, where our proposed work is the used scheduler. Following are three modified algorithms to be used to solve the problem of tasks scheduling and integrated essentially in the cloud broker. We iterate the first and second algorithms based on the total number of tasks. The third algorithm was iterated depending on the total number of tasks and the total number of VMs.

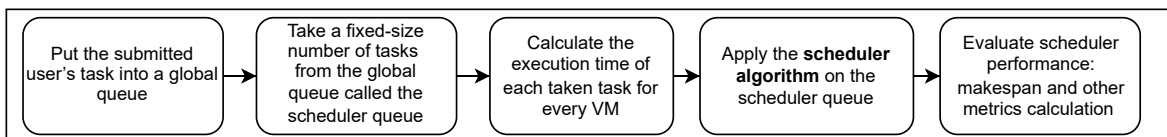


Figure 4. Workflow of the proposed method

#### 2.3.1. First proposed algorithm

The first proposed modified bin packing algorithm (MBPTS-1) begins with sorting all tasks in descending order and then placing each task in the right bin based on algorithm 1. To understand the MBPTS-1 algorithm, we consider six tasks and three VMs with the same processing power as shown in Figure 5, where  $i$  is the number of iterations. We iterate over each task in the ordered list and place the selected task in the next VM based on its defined order  $\{1, 2, 3\}$ . Each time we get to the end of the VMs list, we set the counter back to the beginning of the VMs list until we finish all given tasks in the queue. We sum the execution times of tasks

in each VM and get the biggest value. Figure 5 shows that VM3 has the biggest bin value (makespan) with a total length of  $7; 2+5=7$ .

---

#### Algorithm 1 Pseudo-code of MBPTS-1

---

```

Input:  $ET$  // List of tasks' execution times
Input:  $N$  // Number of bins, Also number of VMs
Output:  $makespan$  // The makespan
Function MBPTS1 ( $ET, N$ ):
   $ET \leftarrow descSort(ET)$  // Sort execution times in descending order
   $binsData \leftarrow (0, \dots, 0)$  // List of bins size initialized with zeros
  for  $i = 0; i < size(ET); i = i + 1$  // Iterate over execution times
  do
     $j \leftarrow i \bmod N$  // j is for iterating over bins data
     $binsData[j] \leftarrow binsData[j] + ET[i]$  // Sum with previous data
  end
   $makespan \leftarrow \max(binsData)$  // Find the biggest value in binsData
  return  $makespan$ 
End Function

```

---

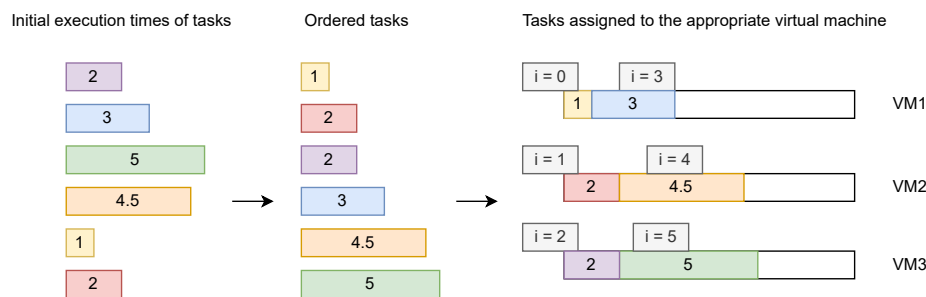


Figure 5. MBPTS-1 example

#### 2.3.2. Second proposed algorithm

The second proposed modified bin packing algorithm (MBPTS-2) begins with sorting all tasks in descending order. Then, the tasks are placed in the bin in reversed order each time the last container arrives (refer to algorithm 2). For the MBPTS-2 algorithm, we apply the same dataset in the example of the first proposition. Figure 6 shows an example of the MBPTS-2 algorithm. We iterate over all ordered tasks in the queue based on the index  $i$ . First, we place each selected task from the queue in the next VM based on its defined order  $\{1, 2, 3\}$ . Each time we reach the beginning or the ending of the VMs list, we reverse the iteration order over VMs. The first three tasks are placed in VM1, VM2 and VM3, respectively. The following three tasks are placed in VM3, VM2 and VM1, respectively, and we reverse the selected VM order in each following three given tasks. Figure 6 shows that the VM2 has the most considerable bin value (makespan) with a total length of  $6.5; 2+4.5=6.5$ .

#### 2.3.3. Third proposed algorithm

The third proposed modified bin packing algorithm (MBPTS-3) begins with sorting all tasks in descending order and then placing each task in the right bin based on the bin's maximum space (refer to algorithm 3 in Appendix). Figure 7 shows an example of the MBPTS-3 algorithm considering the same dataset in the first proposition. At this time, we place each task based on the maximum capacity  $C_{max}$ , which we calculated in (4). In the example the max capacity is  $(2 + 3 + 5 + 4.5 + 1 + 2) / 3 = 5.83$ . For each iteration  $i$ , we iterate over all given VMs to find the maximum available space and place the task into it. The available space of a VM is the  $C_{max}$  minus the execution time of a VM in the current iteration  $i$ . If two or more maximum available spaces are equals, we select the VM based on its defined order. Finally, we sum the execution times of tasks in each VM and get the biggest value. The VM1 has the biggest bin value, with a total length of  $8; 1 + 2 + 5 = 8$  as shown in Figure 7.

**Algorithm 2** Pseudo-code of MBPTS-2

```

Input:  $ET$  // List of tasks' execution times
Input:  $N$  // Number of VMs
Output:  $makespan$  // The makespan
Function MBPTS2 ( $ET, N$ ):
     $ET \leftarrow descSort(ET)$  // Sort execution times in descending order
     $binsData \leftarrow (0, \dots, 0)$  // List of bins size initialised with zeros
     $reverseOrder \leftarrow False$  // A variable to reverse order of iterations over bins
    for  $i = 0; i < size(ET); i = i + 1$  // Iterate over execution times
    do
         $k \leftarrow i \bmod N$  // k is for iterating over bins data
         $j \leftarrow k$ 
        if  $reverseOrder$  // Check if the order is reversed
        then
             $j \leftarrow k$ 
            if  $k = 0$  then
                 $j \leftarrow (N - 1) - k$ 
                 $reverseOrder \leftarrow False$ 
            end
        else
             $j \leftarrow (N - 1) - k$ 
            if  $k = 0$  then
                 $j \leftarrow k$ 
                 $reverseOrder \leftarrow True$ 
            end
        end
         $binsData[j] \leftarrow binsData[j] + ET[i]$  // Sum with previous data
    end
     $makespan \leftarrow max(binsData)$  // Find the biggest value in binsData
    return  $makespan$ 
End Function
    
```

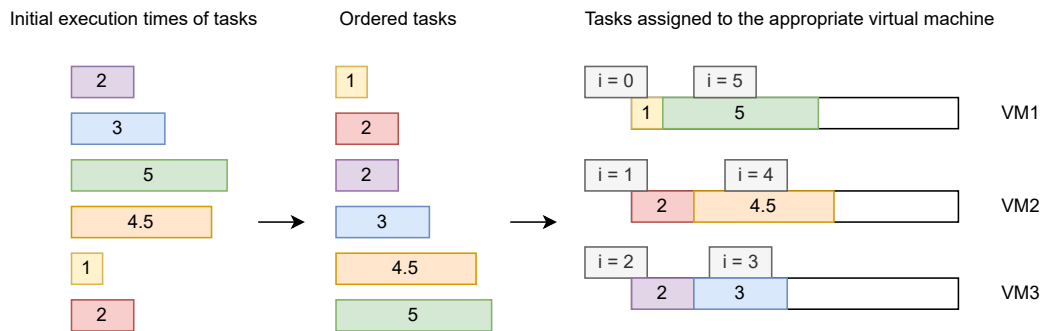


Figure 6. MBPTS-2 example

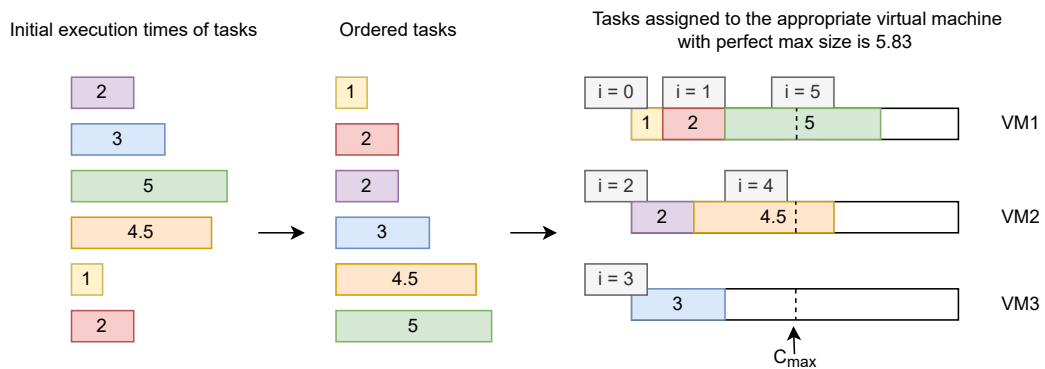


Figure 7. MBPTS-3 example

### 3. RESULTS AND DISCUSSION

#### 3.1. Experimental setup

A simulation was created using the CloudSim simulator [29] to test the proposed implementation against other algorithms. CloudSim tasks (cloudlets) are defined according to their specifications, such as its utilisation model, the number of processing elements, and their processing power in millions of instructions (MI). Cloudsim employs space-shared and time-shared policies for job units scheduling [29]. In this study, the time-shared policy was used to execute different tasks instantaneously within the same host. The simulation was a part of the cloud broker and had the characteristics as presented in Table 1.

Table 1. Resource parameters

Entity Type	Parameter	Value
Data center	Number of data centers	1
Host	Number of hosts	2
	Storage	1 TB
	Memory	2 GB
	Scheduling algorithm	time-shared policy
	Bandwidth	10 GB/s
	MIPS	27 079-177 730
	Number of cores	2-6
VM	Number of VMs	5
	Bandwidth	1 GB/s
	Memory	0.5 GB
	MIPS	9.726
Workload	Source	The HPC2N Seth log
	Numbers of tasks	20-200

The characteristics are defined by CloudSim Feitelson *et al.* [30] example 6 of the provided source code, and the tasks are taken from the generated standard formatted workload "high performance computing center north (HPC2N) Seth Log". Following example 6 and the used workload facilitate the reproducibility of the results of our proposed work. Furthermore, to compare our proposed algorithms, we used the PSO algorithm [31] as a popular algorithm used by many studies in the task scheduling field. We also used The FCFS, the default scheduling algorithm used in CloudSim specified in [29]. The original PSO implementation is not developed for particular optimisation issues like task scheduling. Therefore, a binary version of the PSO algorithm was proposed [32]. The PSO implementation for task scheduling proposed in Figure 2 is used based on the parameters described in Table 2. The experiments were done in a CPU Intel(R) Core(TM) i7-6500U and coded in Java language.

Table 2. PSO used parameters

Parameter	Value
Inertia weight (w)	0.9
Local weight (c1)	1.49445
Local weight (c2)	1.49445
Number of iterations	1000
Number of particles	500

#### 3.2. Experiments and results analysis

The presented algorithms' efficiency was assessed by repeatedly executing those algorithms using different parameters and independent cloudlets. Their performance was evaluated based on the makespan of many sets of tasks. In PSO, Al-Olimat *et al.* [33] run the simulation 100 times and compute the average makespan; our proposed algorithms need to run only once to compute the makespan. Figure 8 shows the results of executing the three proposed algorithms, PSO, FCFS and the perfect makespan calculated in (4), which we are trying to approach to its value. This execution is evaluated in terms of the makespan. We increase the number of tasks in each execution to prove the suggested algorithms' efficacy. As illustrated in Figure 8, the makespan increases with the number of tasks. The makespan of the three proposed algorithms was observed to be better than the PSO and FCFS algorithms, with MBPTS-2 taking the lead. The MBPTS-2 also had a near-perfect makespan. Taking the makespan as an objective of our proposed work helped us in the improvement of other metrics such as the average waiting time, the average resource utilisation, and the DI:

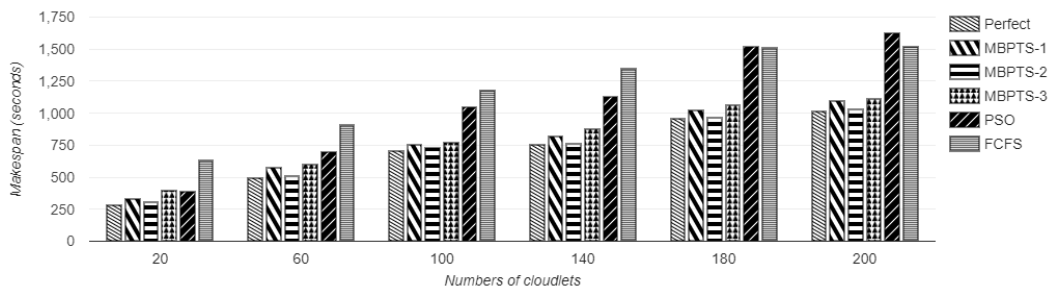


Figure 8. Makespan in seconds with a different number of cloudlets

### 3.2.1. Average waiting time of tasks

The cloudlets were queued and executed based on scheduling algorithms, in which the waiting time algorithm was used to calculate all cloudlet sequences' waiting times and the average waiting time as seen in (5), where  $WT_i$  is the waiting time of the cloudlet  $i$ , and  $n$  is the queue's length.

$$WT_{avg} = \frac{\sum_{i=1}^n WT_i}{n} \quad (5)$$

The outcomes of executing the three suggested algorithms, PSO and FCFS, in terms of the average waiting period are seen in Figure 9. As shown in Figure 9, the proposed algorithms provide optimised solutions that can increase the speed and efficiency in managing the cloudlets' queue by reducing waiting time and queue length, with MBPTS-2 taking the lead in waiting time.

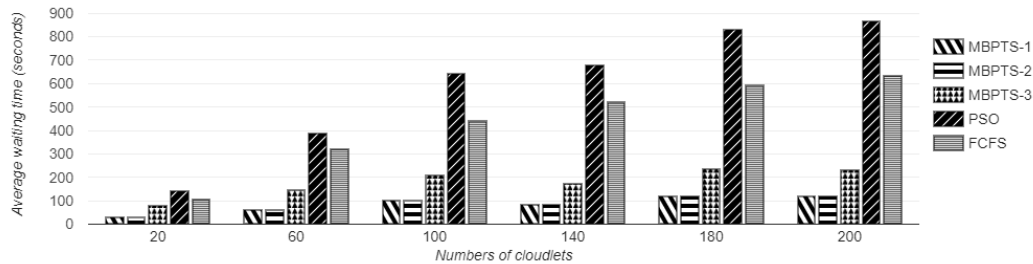


Figure 9. Average waiting time in seconds with a different number of cloudlets

### 3.2.2. Average degree of imbalance (DI)

The degree of imbalance calculates the imbalance among VMs and is an essential QoS metric in proving task allocation efficiency and load balancing between VMs. In this study, it is calculated using (6),

$$DI = \frac{ET_{max} - ET_{min}}{ET_{avg}} \quad (6)$$

where,  $ET_{min}$ ,  $ET_{max}$ ,  $ET_{avg}$  correspond to the minimum, maximum, and average execution times of all VMs, respectively. The scheduling issue's objective aims to reduce DI as a low DI represents a more balanced system [34].

The results of implementing the three suggested algorithms, FCFS, and PSO, in terms of DI are seen in Figure 10. As shown in Figure 10, the proposed modified bin packing algorithms can achieve good load balance and reduce the time to execute tasks, making them superior to the PSO and FCFS algorithms. It is also observed that MBPTS-2 has the best average DI.

### 3.2.3. Average resource utilisation (ARU)

The average resource utilisation (ARU) is essential in task scheduling as high utilisation of resources is desirable. It is calculated using 7 [23],

$$ARU = \frac{\sum_{i=1}^n Ex_{VM_i}}{Makespan \times N} \quad (7)$$



where,  $E_{xVM_i}$  is the duration taken by the  $VM_i$  to complete every given cloudlet, and  $N$  is the total number of VMs. Figure 11 demonstrates the execution of the three proposed algorithms against the PSO and the FCFS algorithms regarding resource utilisation.

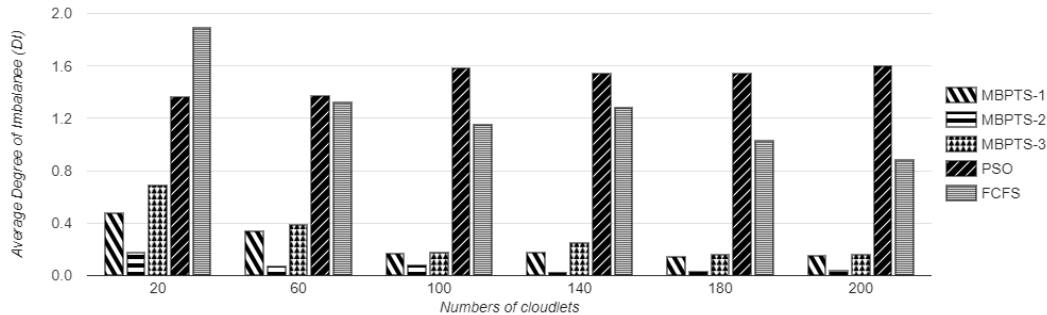


Figure 10. Average DI with a different number of cloudlets

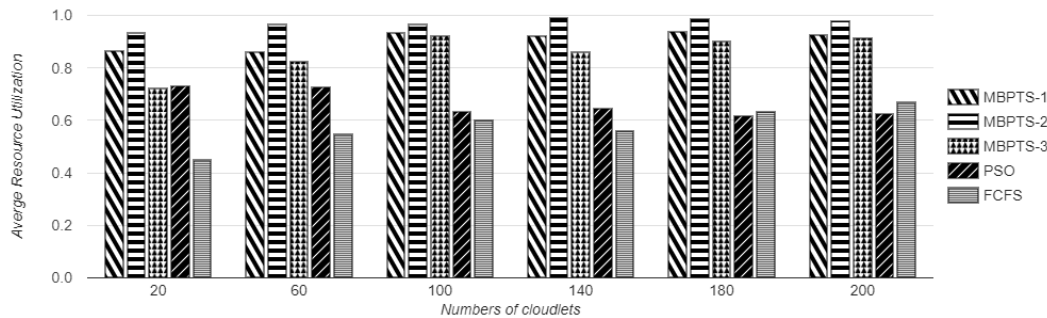


Figure 11. Average resource utilisation with a different number of cloudlets

Figure 11 demonstrates that the proposed algorithms outperform PSO and FCFS algorithms in terms of resource utilisation as the resources are occupied while scheduling tasks. Such quality is valuable for service providers in acquiring maximum profit by renting limited resources. It is also observed that MBPTS-2 has the best average resource utilisation.

To monitor the efficiency of our proposed work much more, especially MBPTS-2, which is taking the lead, we calculate the percentage improvement of each set of tasks for the proposed MBPTS-2 versus PSO and FCFS in terms of makespan, DI, RU, and waiting time. The results are presented in Table 3. We notice that the proposed MBPTS-2 has shown a decrease in the makespan, DI and waiting time and an increase in resource utilisation metric, proving that the proposed MBPTS-2 is successful. We can also see the difference between every percentage in each set of tasks due to the variation of the task lengths. To summarise, the three proposed algorithms outperformed the PSO and the FCFS in terms of makespan, our primary objective, and optimised better the others metrics such as the waiting time, resource utilisation, and degree of imbalance. Furthermore, unlike the method presented in paper [33], where Al-Olimat *et al.* run the simulation 100 times and report the average makespans, our proposed method only needs to be run one time.

Table 3. Percentage improvement (+: increase, -: decrease)

Cloudlets	MPBTS-2 vs PSO (%)						MPBTS-2 vs FCFS (%)					
	20	60	100	140	180	200	20	60	100	140	180	200
Makepsan	- 20.97	- 25.98	- 30.11	- 32.24	- 36.10	- 36.24	- 51.75	- 43.25	- 37.66	- 43.47	- 35.87	- 31.78
WT <sub>avg</sub>	- 78.77	- 83.67	- 83.74	- 87.16	- 85.45	- 85.91	- 71.38	- 80.26	- 76.19	- 83.37	- 79.60	- 80.74
DI	- 86.70	- 94.44	- 95.05	- 98.30	- 98.07	- 97.51	- 90.44	- 94.23	- 93.20	- 97.95	- 97.10	- 95.50
ARU	+ 27.6	+ 33.22	+ 52.97	+ 53.29	+ 59.93	+ 56.77	+ 107.27	+ 76.22	+ 60.41	+ 76.9	+ 55.94	+ 46.58

#### 4. CONCLUSION

Efficient task scheduling is considered one of the major cloud services concerns. An effective scheduler is needed to improve the task scheduling metrics when it comes to larger tasks size. A Better task scheduling ameliorates response time, minimises power consumption and processing time, enhances makespan and throughput, and increases profit by reducing operating costs and raising system reliability. This research presents an efficient task scheduling implementation via the bin packing problem in cloud computing by proposing three modified bin packing algorithms proposed to improve the task scheduling problem and its optimisation metrics. The simulations results demonstrate that the proposed algorithms can minimise waiting time, reduce makespan, and increase resource utilisation. The proposed algorithms also consider load balancing while distributing cloudlets to available resources, an added advantage compared to PSO and FCFS algorithms. As a limitation, this work focused only on makespan, waiting time, resource utilisation and degree of imbalance metrics. Future works may integrate other optimisation methods and consider more quality metrics, such as the migration of tasks between queues, the VM migration concept, and energy consumption.

#### APPENDIX

---

##### Algorithm 3 Pseudo-code of MBPTS-3

---

```

Input:  $ET$  // List of tasks' execution times
Input:  $B$  // List of bins, each bin with a max capacity calculated in (4)
Output:  $makespan$  // The makespan
Function MBPTS3 ( $ET, B$ ):
     $ET \leftarrow descSort(ET)$  // Sort execution times in descending order
     $binsData \leftarrow (0, \dots, 0)$  // List of bins size initialised with zeros
    for  $i = 0; i < size(ET); i = i + 1$  // Iterate over execution times
    do
         $maxSizeIndex \leftarrow -1$  // Index of the large bin
         $maxTarget \leftarrow -1$  // Max Remaining space of all bins
        for  $j = 0; j < size(B); j = j + 1$  // Iterate over bins
        do
             $binSize = B[j]$ 
             $availableTarget = binSize - binsData[j]$ 
            if  $ET[i] \leq availableTarget$  // Check if space available
            then
                 $maxSizeIndex \leftarrow -1$ 
                 $binsData[j] \leftarrow binsData[j] + ET[i]$ 
            else
                if  $availableTarget > maxTarget$  // Find max available space
                then
                     $maxTarget \leftarrow availableTarget$ 
                     $maxSizeIndex \leftarrow j$ 
                end
            end
        end
        if  $maxSizeIndex \neq -1$  // Assign the task to a more available VM
        then
             $maxTarget \leftarrow availableTarget$ 
             $binsData[maxSizeIndex] \leftarrow binsData[maxSizeIndex] + ET[i]$ 
        end
    end
     $makespan \leftarrow \max(binsData)$  // Find the biggest value in binsData
    return  $makespan$ 
End Function

```

---

#### REFERENCES




- [1] P. Mell and T. Grance, "The nist definition of cloud computing," National Institute of Standards and Technology, 2011, doi: 10.6028/nist.sp.800-145.
- [2] H. B. Alla, S. B. Alla, and A. Ezzati, "A novel architecture for task scheduling based on dynamic queues and particle swarm optimization in cloud computing," in *2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech)*, 2016, pp. 108–114, doi: 10.1109/CloudTech.2016.7847686.
- [3] A. Kumar and B. Alam, "Energy harvesting earliest deadline first scheduling algorithm for increasing lifetime of real

- time systems,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, pp. 539-545, 2019, doi: 10.11591/ijece.v9i1.pp539-545.
- [4] S. Ouham and Y. Hadi, “Enhancement in resource allocation system for cloud environment using modified grey wolf technique,” *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 20, no. 3, pp. 1530–1537, 2020, doi:10.11591/ijeecs.v20.i3.pp1530-1537.
- [5] M. A. Aziz and I. H. Ninggal, “Scalable workflow scheduling algorithm for minimizing makespan and failure probability,” *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 8, no. 1, pp. 283–290, 2019, doi: 10.11591/eei.v8i1.1436.
- [6] M. A. Aziz, J. H. Abawajy, and M. Chowdhury, “Scheduling workflow applications with makespan and reliability constraints,” *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 12, no. 2, pp. 482–488, 2018, doi: 10.11591/ijeecs.v12.i2.pp482-488.
- [7] Z. Zhou, J. Chang, Z. Hu, J. Yu, and F. Li, “A modified PSO algorithm for task scheduling optimization in cloud computing,” *Concurrency and Computation: Practice and Experience*, vol. 30, no. 24, 2018, doi: 10.1002/cpe.4970.
- [8] W. Saber, W. Moussa, A. M. Ghuniem, and R. Rizk, “Hybrid load balance based on genetic algorithm in cloud environment,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2477-2489, 2021, doi:10.11591/ijece.v11i3.pp2477-2489.
- [9] S. Harun and M. F. Ibrahim, “A genetic algorithm based task scheduling system for logistics service robots,” *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 8, no. 1, pp. 206–213, 2019, doi:10.11591/eei.v8i1.1437.
- [10] J. Ma, W. Li, T. Fu, L. Yan, and G. Hu, “A novel dynamic task scheduling algorithm based on improved genetic algorithm in cloud computing,” in *Wireless Communications, Networking and Applications*, Springer, 2016, pp. 829–835.
- [11] D. Gabi, A. S. Ismail, A. Zainal, and Z. Zakaria, “Solving task scheduling problem in cloud computing environment using orthogonal taguchi-cat algorithm,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 3, pp. 1489-1497, 2017, doi: 10.11591/ijece.v7i3.pp1489-1497.
- [12] H. B. Alla, S. B. Alla, A. Touhafi, and A. Ezzati, “A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment,” *Cluster Computing*, vol. 21, no. 4, pp. 1797–1820, 2018, doi: 10.1007/s10586-018-2811-x.
- [13] A. S. Abdalkafor and K. M. A. Alheeti, “A hybrid approach for scheduling applications in cloud computing environment,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 1, no. 2, pp. 1387–1397, 2020, doi: 10.11591/ijece.v10i2.pp1387-1397.
- [14] A. M. Alsmadi *et al.*, “Fog computing scheduling algorithm for smart city,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2219-2228, 2021, doi: 10.11591/ijece.v11i3.pp2219-2228.
- [15] N. Edward and J. Elcock, “Task scheduling in heterogeneous multiprocessor environments an efficient aco-based approach,” *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 10, no. 1, pp. 320–329, 2018, doi: 10.11591/ijeecs.v10.i1.pp320-329.
- [16] P. Krishnadoss and P. Jacob, “Ocsa: task scheduling algorithm in cloud computing environment,” *International Journal of Intelligent Engineering and Systems*, vol. 11, no. 3, pp. 271–279, 2018, doi: 10.22266/ijies2018.0630.29.
- [17] R. Khorsand and M. Ramezanpour, “An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing,” *International Journal of Communication Systems*, vol. 33, no. 9, 2020, doi: 10.1002/dac.4379.
- [18] S. Potluri and K. S. Rao, “Optimization model for qos based task scheduling in cloud computing environment,” *International Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 18, no. 2, pp. 1081–1088, 2020, doi: 10.11591/ijeecs.v18.i2.pp1081-1088.
- [19] Y. A. G. Alyoubaki and M. F. Al-Rawi, “Novel load balancing approach based on ant colony optimization technique in cloud computing,” *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 10, no. 4, pp. 2320–2326, 2021, doi: 10.11591/eei.v10i4.2947.
- [20] A. Chraibi, S. B. Alla, and A. Ezzati, “A novel artificial intelligence technique for cloud computing using a new heuristic initialisation and PSO-parallel execution,” *Proceedings of the Future Technologies Conference (FTC) 2021*, vol. 3, oct 2021, pp. 362–376, doi: 10.1007/978-3-030-89912-7\_28
- [21] A. Chraibi, S. Ben Alla, and A. Ezzati, “Makespan optimisation in cloudlet scheduling with improved dqn algorithm in cloud computing,” *Scientific Programming*, vol. 2021, 2021, doi: 10.1155/2021/7216795.
- [22] A. Karthick, E. Ramaraj, and R. G. Subramanian, “An efficient multi queue job scheduling for cloud computing,” in *2014 World Congress on Computing and Communication Technologies*, 2014, pp. 164–166, doi: 10.1109/WCCCT.2014.8.
- [23] M. Kalra and S. Singh, “A review of metaheuristic scheduling techniques in cloud computing,” *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 275–295, 2015, doi: 10.1016/j.eij.2015.07.001.
- [24] A. R. Brown, *Optimum packing and depletion*, Macdonald, 1971.
- [25] M. R. Garey and D. S. Johnson, *Computers and intractability*, W. H. Freeman and Company, 1979, vol. 174.
- [26] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*, John Wiley and Sons, 1990.
- [27] D. Du and P. M. Pardalos, *Handbook of combinatorial optimization*, Springer Science and Business Media, 1998,




- vol. 4.
- [28] R. Yesodha and T. Amudha, "A comparative study on heuristic procedures to solve bin packing problems," *International Journal in Foundations of Computer Science and Technology*, vol. 2, no. 6, pp. 37–49, 2012, doi: 10.5121/ijfst.2012.2603.
- [29] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011, doi: doi.org/10.1002/spe.995.
- [30] D. G. Feitelson, D. Tsafir, and D. Krakov, "Experience with using the parallel workloads archive," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2967–2982, 2014, doi: 10.1016/j.jpdc.2014.06.013.
- [31] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," in *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002, doi: 10.1109/4235.985692.
- [32] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, 1997, pp. 4104–4108, vol. 5, doi: 10.1109/ICSMC.1997.637339.
- [33] H. S. Al-Olimat, M. Alam, R. Green and J. K. Lee, "Cloudlet scheduling with particle swarm optimization," *2015 Fifth International Conference on Communication Systems and Network Technologies*, 2015, pp. 991–995, doi: 10.1109/CSNT.2015.252.
- [34] K. Li, G. Xu, G. Zhao, Y. Dong and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," *2011 Sixth Annual Chinagrid Conference*, 2011, pp. 3–9, doi: 10.1109/ChinaGrid.2011.1

## BIOGRAPHIES OF AUTHORS






**Amine Chraibi**    is a Ph.D. student at the Faculty of Science and Technology, Department of Mathematics and Computer Science, Hassan First University, Settat, Morocco. He obtained his Master of Science and Technology degree in Networks and Computer Systems from Faculty of Science and Technology in 2016. His researches are in Task scheduling in cloud computing, heterogeneous computing and artificial intelligence. He can be contacted at email: a.chraibi@uhp.ac.ma.



**Said Ben Alla**    is a Professor at ENSA Berrechid, Department of Mathematics and Computer Science, Hassan First University, Settat, Morocco. He received his Master of Science degree in Telecommunications and Networks in 2009 from the University of Cadi Ayyad, Morocco and his Ph.D. from Faculty of Sciences and Techniques (FSTS), Hassan First University of Settat, Morocco in 2013. His researches are in cloud computing, wireless ad hoc, wireless sensor networks (WSNs), and embedded real-time systems. He can be contacted at email: saidb\_05@hotmail.com.



**Abdellah Ezzati**    is a Professor at the Faculty of Sciences and Techniques, Department of Mathematics and Computer Science, Hassan First University, Settat, Morocco. In 2012 he was awarded a Research Habilitation degree from the First University of Hassan, Faculty of Science and Technology (FSTS), Settat, Morocco. His researches are in protocol specifications, mobility Management, distributed systems, cloud computing, and wireless sensor network (WSN) management. He can be contacted at email: abdezzati@gmail.com.