WGOA: Whale-Genetic Optimization Algorithm for Energy Efficient Task Scheduling in Cloud Computing

Conference Paper · August 2018

CITATIONS
0

READS
210

2 authors:

Garg Ritu
National Institute of Technology, Kurukshetra
88 PUBLICATIONS **1,027** CITATIONS
SEE PROFILE

Mohan Sharma
National Institute of Technology, Kurukshetra
2 PUBLICATIONS **22** CITATIONS
SEE PROFILE

WGOA: Whale-Genetic Optimization Algorithm for Energy Efficient Task Scheduling in Cloud Data Center

Mohan Sharma[1] and Dr. Ritu Garg[2]

[1, 2] Computer Engineering Department,
National Institute of Technology, Kurukshetra 136119, India
mohan@mohansharma.me[1] ritu.59@nitkkr.ac.in[2]

Abstract. Energy efficient task scheduling in heterogeneous cloud environment is now a days challenging problem and it belongs to NP-class problems. In this paper, we focused on minimizing energy consumption and improving performance (makespan) while reducing execution overhead for independent task scheduling. To achieve the same, we proposed a hybrid whale-genetic optimization algorithm to balance those two objectives; makespan and energy consumption. Our proposed algorithm also aims to reduce the execution time by reducing the amount of iterations spend in local optimal region. Extensive simulation has been carried out and compared over original Whale Optimization Algorithm and well-known MinMin algorithm. Results clearly manifest that our proposed algorithm out performs original whale optimization algorithm as well as MinMin heuristic algorithm and reduced execution overhead up to more than half of what required originally.

Keywords: cloud data center; independent task scheduler; whale optimization algorithm; genetic algorithm; hybrid multi-objective optimization; execution overhead; energy efficiency; makespan;

1 Introduction

Increasing demands of high performance computing (HPC) resources led to the new computing model known as Cloud Computing. These resources are served to end users just like other utility services such as electricity etc along with other services such as Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS). HPC resources comes with huge energy requirements which increases maintenance and energy related costs.

Energy consumption in a data center mainly consists of computing energy, communication energy, cooling energy and storage energy. The approximate peak power distribution in a data center and shows that computing elements (42%) and cooling elements (14.3%) consumes more than 50% of total energy consumption [1]. It is predicated that energy consumption by Information and Communication Technology (ICT) will increase to 14% before 2021, currently it makes up 8% of global energy consumption [2]. So there is a huge demand of energy efficient technologies in cloud computing.

Task allocation can result into poor resource utilization with huge energy consumption and can also reduce the resource' life span. In fact, it is the well-known problem of task scheduling in cloud computing and considered as NP-class problem. Energy efficient task scheduling is a decision making process which choses suitable HPC resources for a set of tasks considering resulting performance and energy consumption.

Recently, researchers targeting the energy efficient task scheduling in cloud computing and proposed many techniques to reduce the energy consumption. Dynamic voltage and frequency scaling (DVFS) [3] [13], SpeedStep [4] based scheduling provides the energy efficiency by adjusting operating frequencies and voltages according to current workload. To generate optimal schedule, meta-heuristic algorithms such as Genetic Algorithm (GA) [5] [6], Particle Swarm Optimization (PSO) [7], Ant Colony Optimization (ACO) [9] and Whale Optimization Algorithm [8] are widely used for the purpose of optimizing the task schedule. While, only few of them addresses energy efficient task scheduling problem. One of the problem with meta-heuristic based approaches is that they relatively takes more time than the simple algorithm [11].

In our work, we are considering a problem to schedule set of independent tasks to set of virtual machines in IaaS cloud environment as a multi-objective optimization problem with the aim to optimize makespan and energy consumption while reducing execution overhead. To achieve same goal, we have proposed a new hybrid meta-heuristic algorithm Whale-Genetic Optimization Algorithm (WGOA) which combines the WOA' better search mechanism along with GA' exploration capability.

2 Environment Model

In this section, we discuss the mathematical system model presenting how cloud environment works, task scheduling model presents the how tasks and scheduler defined and energy consumption model which describes how we estimated energy consumption for a task schedule.

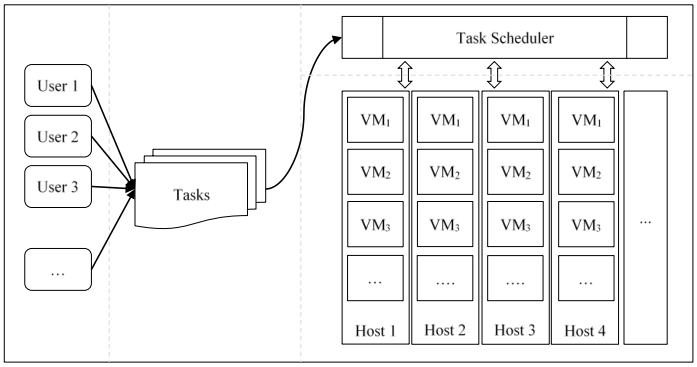

Fig. 1. System model

2.1 System Model

Figure 1 presents system model where the users submits different tasks to cloud data center via a fronted task management panel. Those tasks will scheduled on available virtual machines by a scheduler. Upon task completion, task results will be sent back to the end users. This system model is modelled as *SM*, which basically is a set of physical hosts $P_1, P_2, P_3 \ldots P_N$ and its connectivity.

$$SM = \{P_1, P_2, P_3 \ldots P_N\} \tag{1}$$

Physical hosts P_i are may be grouped together or placed in table order. So that, each physical host can communicate with central entity manager i.e. Virtual Machine Manager (VMM). We defined physical host P_i as follows:

$$P_i = (Speed_i, Power_i^u) \tag{2}$$

Where, $Speed_i$ defines the execution speed in terms of millions of instructions per second (MIPS). $Power_i^u$ defines the energy consumption of P_i on u utilization as per Figure 2. Each physical host P_i is having multiple virtual machines on it to reduce the amount of hardware in use and it is defined as follows:

$$V_{ij} = \left(Speed_{ij}\right) \tag{3}$$

Where, $Speed_{ij}$ defines the execution speed of j^{th} virtual machine deployed on i^{th} physical host. Host utilization is calculated by dividing the total utilized virtual capacity by the total physical host capacity, which is defined as follows:

$$U_i(t) = \frac{\sum_{V_{ij} \in P_i}(speed\ utilized\ by\ V_{ij}\ at\ time\ t)}{Speed_i} \tag{4}$$

Where, U_i defines the host utilization of i^{th} physical host at time t.

2.2 Task Scheduling Model

Multiple users in parallel submits many tasks at a time. Such set of independent tasks *TS* can be defined as follows:

$$TS = \{T_1, T_2, T_3 \ldots T_k\} \tag{5}$$

where, T_k represents a task. Task is a unit of workload expressed as Million Instructions (MI) and it is defined as follows:

$$T_k = (W_k) \tag{6}$$

where, W_k represents the workload of k^{th} task. Scheduler *S* maps the available resources to the tasks by producing a task and virtual machine pair. It is defined as follows:

$$S(TS, VM) = \{ \ldots T_k \rightarrow V_{ij} \ldots \} \tag{7}$$

where, *TS* is a set of independent tasks, *VM* is a set of available virtual machines and $T_k \rightarrow V_{ij}$ shows that k^{th} task is scheduled on j^{th} virtual machine deployed on i^{th} physical host.

In this work, we optimizes makespan (performance) along with energy consumption, makespan is a maximum finish time of a task among the input task set and it is defined as follows:

$$makespan = \max_{\forall V_{ij} \in S} FT(V_{ij}) \tag{8}$$

where, V_{ij} represents every selected virtual machine by a scheduler S and $FT(V_{ij})$ gives us a time at which V_{ij} finishes all the allocated tasks. Finish time (FT) of V_{ij} virtual machine is defined as follows:

$$FT(V_{ij}) = \sum_{\forall T_k \in V_{ij}} EET(T_k, V_{ij}) \tag{9}$$

where, T_k is the task allocated to V_{ij} virtual machine and $EET(T_k,V_{ij})$ gives us the estimated execution time of task T_k on V_{ij} virtual machine and it is defined as follows:

$$EET(T_k, V_{ij}) = \frac{W_k}{Speed_{ij}} \tag{10}$$

where, W_k is workload of k^{th} task, and $Speed_{ij}$ is execution speed of j^{th} virtual machine (deployed on i^{th} physical host).

2.3 Energy Consumption Model

We are considering heterogeneous cloud environment. As the workload on physical machine increases, the energy consumed by it also increases and when physical host is idle then it consumes nominal energy required to run it in idle mode. Energy consumption by physical hosts tends to increase as the amount of workload increases [3].

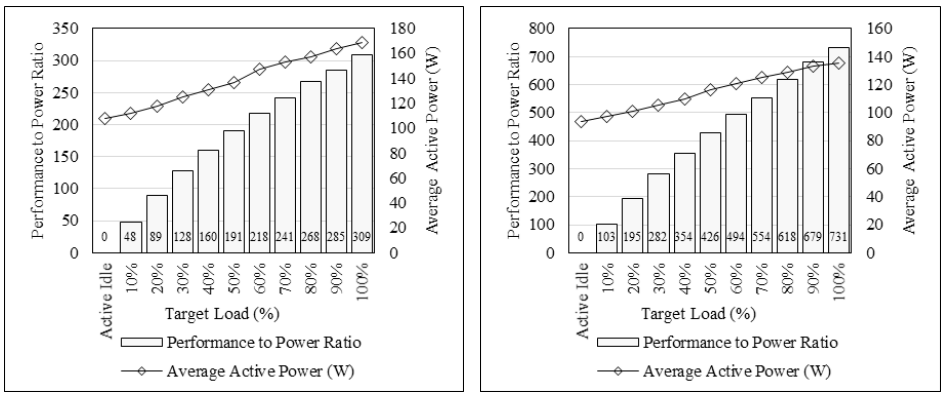

Fig. 2. Correlation between amount of workload and power consumption of HP ProLiant ML110 G5 (left) and HP ProLiant ML110 G4 (right)

Figure 2 shows the correlation between host utilization and energy consumption of two widely used server host machines i.e. HP ProLiant ML110 G5 and ML110 G4. To estimate the energy consumption using the correlation suggested in Figure 2, we

first calculated the energy consumed by the virtual machine for particular task and then using summation of these energy consumption data we defined the total energy consumption. Energy consumed by the task on a virtual machine started its execution at *estarted* is defined as follows:

$$EC(T_k, V_{ij}) = \sum_{time=estarted}^{EET(T_k,V_{ij})} Power_i^{U_i(time)} \tag{11}$$

Using the Eq.11, we can estimate our second objective the energy consumption of entire schedule S as follows:

$$E = \sum_{\forall (T_k \to V_{ij}) \in S} EC(T_k, V_{ij}) \tag{12}$$

3 Proposed WGOA: Whale-Genetic Optimization Algorithm

In this section, we presents the proposed WGOA algorithm for scheduling set of independent tasks with the aim to optimize 1) makespan and 2) energy consumption. We considered WOA [12] as our primary algorithm and it is a nature-inspired optimization algorithm which follows how hump whale sharks hunts and finds their prey using bubble-net hunting method.

Problem with WOA [8] [12] is that for the most of iterations (generations) it stays trapped in local optimal point. To get the WOA out of local optimal point, we use GA. GA is adaptive heuristic search mechanism which simulates and follows the survival of fittest among individual over the number of iterations.

To reduce the execution overhead by eliminating the local optimal point or trap time, we define new parameters which combines the exploration and exploitation capability of WOA and GA respectively. Considered parameters are as follows: 1) *whale_leader_step*; this parameter counts that for how many iterations same whale leader survived, 2) *whale_leader_thresold*; this parameter defines the threshold to indicate that after how many iterations whale leader considered to be trapped in local optimal point, 3) *ga_iterations*; defines the number of iterations GA requires to get WOA out of local trap.

Algorithm 1 presents the pseudo code of proposed WGOA algorithm. At Line 1, it initializes algorithm specific parameters as specified in Table 1 and generates randomly initialized populations of whales (solutions) at Line 2. WGOA will run for the specified number of iterations from Lines 3 to 20. Algorithm 1 operates in two phases: 1) in first phase (Line 4-7), Algorithm 1 moves whales according to WOA (Algorithm 2). After moving whales, we put a check to update the *whale_leader_step* parameter. It updates the parameter by comparing the position of last whale leader and current whale leader if it's happen to be the same position then it increments parameter otherwise skips, 2) in second phase (Line 8-18), if the *whale_leader_step* parameter crosses the threshold value specified by the *whale_leader_thresold* that means WOA is trapped in local optimal point for past *whale_leader_thresold* iterations.

Then, Algorithm 1 executes GA (Algorithm 3) to get the WOA out of local trap. Further, Algorithm 1 combines the best individuals from both the algorithms and replaces worst whale leader by best whale leader (Line 11-16). At Line 17, Algorithm 1 skips *ga_iteration* number of iterations consumed by GA along with extra number of iterations to reduce the execution overhead. WGOA repeats same process to find the optimal solution till the termination condition met.

Algorithm 1 Proposed WGOA

```
1. Initialize algorithm specific parameters
2. Randomly initialize population of whales WOA0.pop
3. Do
4.     WOA(i) = WOAIteration(WOA(i-1).pop)
5.     If WOA(i).LeaderWhale == WOA(i-1).LeaderWhale
6.         whale_leader_step = whale_leader_step + 1
7.     End If
8.     If whale_leader_step >= whale_leader_thresold
9.       whale_leader_step = 0
10.      GA = GeneticEvolution(WOA(i).pop, ga_iterations)
11.      WOA(i).pop = WOA(i).pop U GA.pop
12.      Sort WOA(i).pop as per the fitness value
13.      Drop the worst half whale population
14.      If GA.LeaderWhale >= WOA(i).LeaderWhale
15.        WOA(i).LeaderWhale = GA.LeaderWhale
16.      End If
17.      i = i + ga_iterations + (ga_iterations/2)
18.    End If
19.    i = i + 1
20. While (i < max_iteration)
21. Return WOA(i).LeaderWhale
```

Algorithm 2 represents the original version of WOA algorithm with only single iteration. It takes last whale population as input and moves whales. It first calculates fitness value of each individuals and sets the best whale as whale leader *LeaderWhale*. In next step, other whales proceeds towards the best available whale leader and updates their position using Eq. 13.

$$W_i = LeaderWhale - M \cdot |Q \cdot LeaderWhale - W_{i-1}| \tag{13}$$

where, W_i and W_{i-1} represents new and old position of whale, M and Q represents coefficient vector calculated using Eq. 14.

$$\begin{aligned} M &= 2m \cdot n - m \\ Q &= 2n \end{aligned} \tag{14}$$

where, the value of m decreased from 2 to 0 over iterations and n represents the random vector in [0,1].

In next exploitation step, two methods are selected based on probability variable *pr*. First method is known as shrinking encircling method in which simply the value

of *m* is set to [-1, 1] and new position of whales calculated based whale's current position and leader whale position. Second method is known as spiral updating in which the whale's position is updated in spiral manner using Eq. 15.

$$W_i = |LeaderWhale - W_{i-1}| \cdot h^{st} \cdot \cos(2\pi t) + LeaderWhale \quad (15)$$

where, *s* represents the constant and *t* is the value in [-1,1]. In next exploration step, random whale is selected from the population and its position updated using randomly generated whale.

Algorithm 2 WOAIteration(population)

```
1. Calculate fitness of whale population
2. Set LeaderWhale to best individual
3. Update m, M, Q, t and random probability variable pr
4. If pr < 0.5 then
5.     If |M| < 1 then
6.       Update position of whales using equation 12
7.     Else
8.       Update position of whales using randomly selected
         whale
9.     End If
10. Else
11.    Update position of whales using equation 14
12. End If
13. If any whale goes outside of search range then update its
    position
```

Algorithm 3 presents the GA algorithm. GA first calculates the fitness of each individuals from populations and does the parent selection based using Roulette Wheel Selection method for creating off-springs. Next, it performs crossover operator over selected parents with the probability of P_c followed by mutation operator which also performed with the probability of P_m. Then, in the end, it performs survival selection which selects individuals based on fitness value and best individuals are chosen for the next iteration. This continues till the termination condition met.

Algorithm 3 GeneticEvolution(population, iterations)

```
1. G = 1
2. Do
3.     Calculate the fitness of population
4.     Perform Parent Selection using Roulette Wheel
       Selection
5.     Crossover with probability Pc¬
6.     Mutation with probability Pm
7.     Survivor Selection
8.     G = G + 1
9. While (G < iterations)
10. Set the best individual as LeaderWhale
```

Algorithm 4 presents the fitness function for WGOA algorithm. It also works in two phases. In first phase, fitness function focuses on minimizing makespan value only. In second phase, fitness function focuses on minimizing energy consumption while constraining makespan value to the best value obtained in first phase.

```
Algorithm 4 Fitness(schedule)
1. If current_iteration < max_iteration/2
2.    Calculate the makespan using Eq. 10.
3.  Return makespan
4. End If
5. If current_iteration = max_iteration/2
6.    Calculate the makespan using Eq. 10.
7.    ms = makespan
8. End If
9. If current_iteration >= max_iteration/2
10.   Calculate makespan using Eq. 10.
11.   Calculate energy consumption using Eq. 12.
12.   If makespan > ms
13.     Return Infinite
14.   Else
15.     Return energy consumption
16.   End If
17. End If
```

Table 1. Algorithm specific parameters and its default values.

Parameter Name	Parameter Description and Default Value
i	Iteration counter, 0
P_c	Crossover probability, 0.9
P_m	Mutation probability, 0.1
whale_leader_step	Steps of same leader over generation, 0
whale_leader_thresold	Threshold value up to which same leader is allowed, 10
ga_iteration	Number of iteration required by GA to get WOA out of local trap, 30
max_iteration	Maximum number of iteration, 250
current_iteration	Current number of iteration, equals to i
population_size	Size of population, 30
ms	Makespan value after half of iterations, 0

4 Results and Discussions

This section presents the experimental setup and results of our proposed algorithm WGOA. The experimentation of WGOA is performed over personal computer with Intel Core-i7 with 8GB of RAM using Windows 10 Operating System. WGOA is implemented using MATLAB and performance evaluation is performed using makespan and energy consumption value. Result is compared with the original WOA algorithm and state-of-art MinMin algorithm. For measuring the performance we have considered heterogeneous computing resources with varying computing capacity (1500 to 4500 MIPS) and set of randomly generated independent tasks. We considered default parameter values as specified in Table 1.

First, we have compared the performance of algorithms over 100 tasks with different number of virtual machines (4, 8, 16 and 32) in Figure 3. Figure 3a shows that our proposed algorithm out performs other algorithm in terms of makespan and able to get better makespan value. Similarly, Figure 3b shows energy consumption comparison which also states that our proposed algorithm is able to get better energy savings.

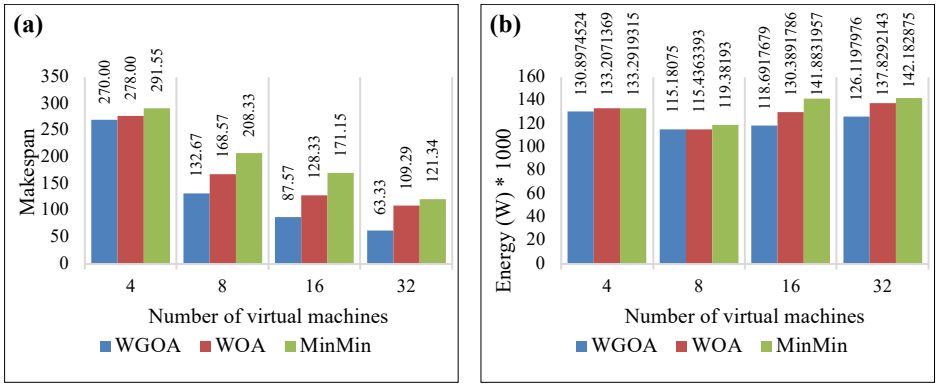

Fig. 3. Experiment result: 100 tasks over 4, 8, 16 and 32 virtual machines a) makespan comparison b) energy consumption comparison

Next we compared the performance of algorithms on 8 virtual machines with varied number of tasks (25, 50, 100 and 1000) and the results are shown in Figure 4. Figure 4 represents similar results and shows that proposed algorithm is stable and out performs other algorithm in terms of makespan as well as energy consumption.

Table 2 shows the execution overhead comparison of proposed algorithms over original WOA algorithm. It clearly manifests that WGOA achieves better energy savings and performance while reducing the execution overhead more than half of what required by the original WOA.

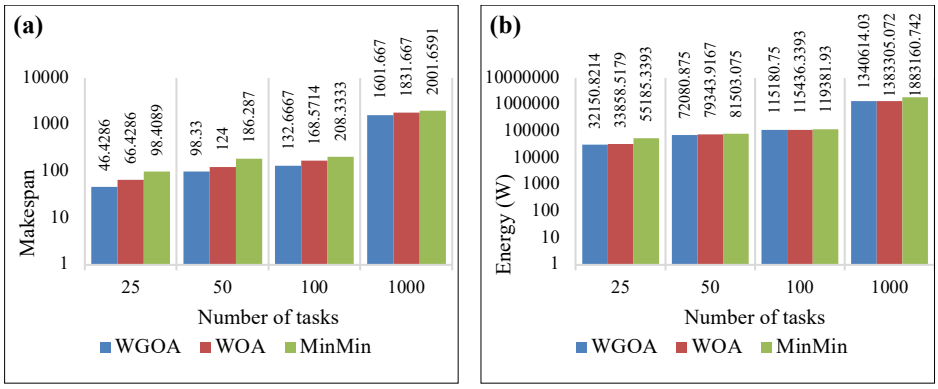

Fig. 4. Experiment result: 25, 50, 100 and 1000 tasks on 8 virtual machines; a) makespan comparison b) energy consumption comparison

Table 2. Execution overhead comparison in seconds.

	100 tasks over 4, 8, 16 and 32 virtual machines				8 virtual machines with 25, 50, 100 and 1000 tasks			
Algorithm	4 VM	8 VM	16 VM	32 VM	25 tasks	50 tasks	100 tasks	1000 tasks
WGOA	0.9887	1.0992	1.3871	1.3994	0.5613	0.6835	1.0231	23.4491
WOA	3.9718	4.0360	4.2992	4.7661	1.7282	2.5055	4.0461	42.6897

5 Conclusion

Problem of energy efficient task scheduling is of prime importance in cloud computing. Higher energy consumption leads to higher managerial cost, electricity cost and also harms nature. In literature many task scheduling algorithms have been developed but only few of them addresses the energy efficiency issue in cloud computing. In our work we proposed energy efficient task scheduler while maintaining the performance and reducing the execution overhead. We proposed a new hybrid algorithm to optimize the energy efficiency while maintaining performance. In our hybrid algorithm, we have improved original whale optimization algorithm with the help of genetic algorithm. Original whale optimization algorithm has a tendency to fall into local optimal region where the genetic algorithm helps it to get out of local trap. Simulation experiments has been carried out over the different number of tasks as well as available virtual machines in heterogeneous environment and results clearly indicates that WGOA out performs other algorithms in terms of makespan as well as energy consumption while reducing the execution overhead up to half of what required by original whale optimization algorithm. In future, we will extend our work to consider temperature effects and real world scientific applications (workflows).

References

1. Clidaras, Jimmy, et al. “The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition.” Synthesis Lectures on Computer Architecture, vol. 8, no. 3, 2013, pp. 1–154, doi:10.2200/s00516ed2v01y201306cac024.
2. E. Gelenbe and Y. Caseau. The impact of information technology on energy consumption and carbon emissions. Ubiquity, 2015(June):1:1–1:15, June 2015
3. R. Buyya, A. Beloglazov, J. Abawajy, Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges, in: Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2010, Las Vegas, USA, 2010.
4. Intel whitepaper 30057701 (2004) Wireless Intel SpeedStep Power Manager: optimizing power consumption for the Intel PXA27x processor family. http://61.108.100.48/%EC%98%A4%ED%94%88%ED%94%8C%EB%9E%AB%ED%8F%BC%EC%8B%A4%EC%8A%B5/DataSheets/BASE_Board/PXA270/White%20Paper/30057701.pdf. (Last accessed April 25, 2018).
5. Casas, I., Taheri, J., Ranjan, R., Wang, L., & Zomaya, A. (2016). GA-ETI: An enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments. Journal of Computational Science. Epub ahead of print. https://doi.org/10.1016/j.jocs.2016.08.007
6. Salido, M.A., Escamilla, J., Giret, A. et al. A genetic algorithm for energy-efficiency in job-shop scheduling, Int J Adv Manuf Technol (2016) 85: 1303. https://doi.org/10.1007/s00170-015-7987-0
7. Yassa, S., Chelouah, R., Kadima, H., & Granado, B. (2013). Multi-Objective Approach for Energy-Aware Workflow Scheduling in Cloud Computing Environments. The Scientific World Journal, 2013, 1-13. doi:10.1155/2013/350934
8. Sreenu, K. & Sreelatha, M. "W-Scheduler: whale optimization for task scheduling in cloud computing", Cluster Comput (2017). https://doi.org/10.1007/s10586-017-1055-5
9. M. Tawfeek, A. El-Sisi, A. Keshk, and F. Torkey, “Cloud task scheduling based on ant colony optimization,” 8th International Conference on Computer Engineering & Systems (ICCES), pp. 64-69, 2013.
10. A. Xu, Y. Yang, Z. Mi and Z. Xiong, "Task Scheduling Algorithm Based on PSO in Cloud Environment", 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), 2015
11. Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY (2008) A survey of scheduling problems with setup times or costs. Eur J Oper Res187(3):985–1032
12. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. 95, 51–67 (2016)
13. Wang, S., Qian, Z., Yuan, J., & You, I. (2017). A DVFS Based Energy-Efficient Tasks Scheduling in a Data Center. IEEE Access, 5, 13090-13102. doi:10.1109/access.2017.2724598