Modified Harmony Search Algorithm for Scheduling Applications in Cloud Environment

Abderrahim BOUCHAIR, Sid Ahmed MAKHLOUF, and Belabbas YAGOUBI

Department of Computer Science, University of Oran1 Ahmed Ben Bella, Oran, Algeria. E-mail: {bouchair.abderrahim,sidahmed.makhlouf,byagoubi}@gmail.com

Abstract. Cloud computing is encountering a fast development in industrial and academical fields, providing on-demand services such as infrastructures and applications whether the resource type is physical or virtual. Data Center Networks (DCNs) architecture reflects directly on its scalability, fault-tolerance and more importantly on the Cloud workflow. For that matter, server-centric data centers are facing a main issue regarding applications scheduling due to traffic forwarding that relies on servers imitating switches. The challenge is to choose an optimization method so that Cloud users constraints are satisfied. If the problem is of small size and reduced complexity, the implementation of an exact method may be sufficient to determine an optimal solution. In the case of massive size problems, the approximate methods are the most efficient way to get as close as possible to the optimal solution. In this paper, we focused on the job-shop scheduling type problem, with Makespan minimization as a criterion. The problem is known NP-hard, we propose for its resolution the meta-heuristic "Harmony Search." After a phase of experimental determinations of the approach parameter values using the CloudSim which is a framework for simulation and modeling of cloud computing, a set of validation tests was carried out on the most known benchmarks. Overall, the results remain encouraging.

Keywords: Cloud computing, Cloud Scheduling, Meta-heuristic, Harmony Search Algorithm (HSA), NetCloudSim.

1 INTRODUCTION

Since its first launch, Cloud computing is proving an evolving resistance and a scalability development over various fields in our practical life. This growth made user access experience similar to any daily life needs like electricity or water. Hence, numerous requests are continuously received. Therefore several resource management policies were adopted to maintain an efficient performance across application scheduling in DCNs. Cloud scheduling can be divided into two categories. When all tasks are independent, it can be assigned to the processors independently without any predefined order of execution. In the case of dependent scheduling (workflows), scheduling is very complicated [1].

A proper task scheduling creates a clear vision about resource availability and helps define the objectives associated. Researchers still face complicated optimization problems that are very difficult to solve. Over the past few years, Cloud computing has adopted many operational research methods like meta-heuristics methods, which are usually iterative stochastic algorithms that progress towards a global optimum. Our work presents an adaptation of a recent meta-heuristic called Harmony Search (HS) for the resolution of the job-shop scheduling in a Cloud-based server-centric architecture. Indeed, this method, which was recently developed by Geem [2] and inspired by the musical improvisation process, intended to solve optimization problems. This paper is organized as follows: In section two, a list of previous related works was provided. Next, section three and four present a brief summary of server-centric architectures in the Cloud and Scheduling problem definition respectively. In the fifth section, a job-shop problem and HSA basics were presented. Our contribution including a representation of implementation and the result of experimentations are described in section six — finally, a conclusion with a recommended future works.

2 Related Works

Many previous studies inspected the consolidation of Cloud-based environment issues with industrial manufacturing problems. However, just a few works were dedicated to solve local tasks scheduling problems. Fathi and khanli [3], employed HSA for virtual machine (VM) consolidation to allow the reduction in energy consumption respecting Service Layer Agreement (SLA) violation and live migrations quantity In [1], a merged HS algorithm was proposed with Group technology aiming to reduce makespan for Cloud environment. Al- maamari and Omara [4] has considered an amalgamation of the Particle Swarm Optimization algorithm and the Cuckoo search (CS) algorithm to resolve the task scheduling problems, using the CloudSim [5] simulator to evaluate the proposed algorithm. A Polyrhythmic HS was suggested by Melnik and Trofimenko [7], which is a new conception of HS for scientific workflow scheduling to find an optimal solution in terms of scheduling execution time. Haoqian and Lianglun [8], has presented a model (ACO-HS) mixing the ant colony optimization algorithm and harmony search algorithm. ACO-HS can effectively retain the high accuracy and parallelism of the ACO algorithm, and combine with the capabilities of fast global search from HSA, evaluated by a simulation using the CloudSim software.

Job-shop scheduling provides multiple advantages such as getting real-time feedback notifications; it can operate different types of resource simultaneously and more importantly, it grants regrouping secondary objectives in one single objective, unlike the previously mentioned works which focus on developing a mechanism to solve one specific constraint. All this has motivated us to implement HSA to reduce the makespan of Cloud application as our primary goal and eventually lowering processing requirements, energy consumption and resource utilization in DCNs by examining several job-shop benchmark problems in the literature.

3 Server-centric Data Center

Today's data centers (DCs) enters span widely in the field of Cloud computing. In server-centric architectures, servers are responsible for networking and routing, whereas commodity switches without modification are used only for forwarding packets [9]. There are three typical server-centric such as Bcube [10], Dcell [11], and Ficonn [12].

3.1 Bcube.

A BCube topology relies on servers to take part in the traffic network, requiring a high bandwidth to support intensive computing application. As shown in figure 1, the benefits of BCube design is that it can provide fault tolerance and load balancing and while requiring lower cooling and manufacturing cost [14].

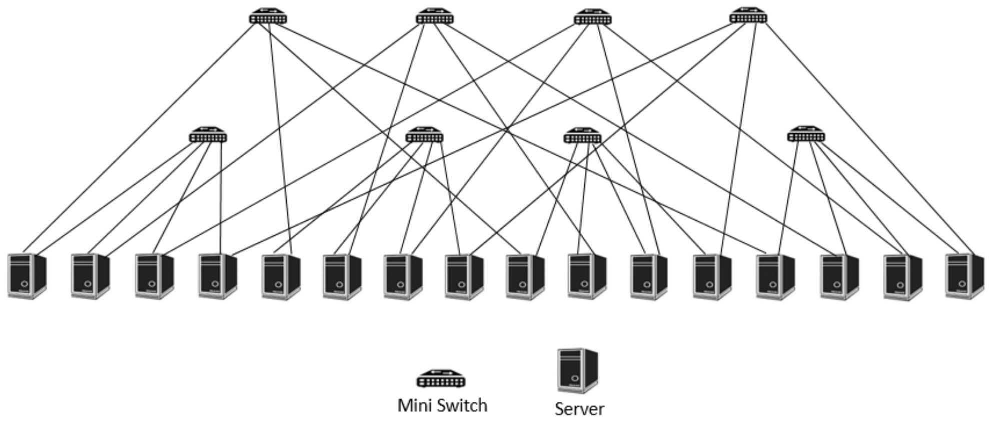


Fig. 1: The BCube topology.

3.2 DCell.

A Dcell is a typical server-centric topology where we can find a direct point to point connectivity between servers as shown in figure 3, which requires a lot of ports. This topology has Dcell$_0$ as an essential element that includes a mini-switch connected to n servers and provide excellent results in fault-tolerance using its routing protocol [15].

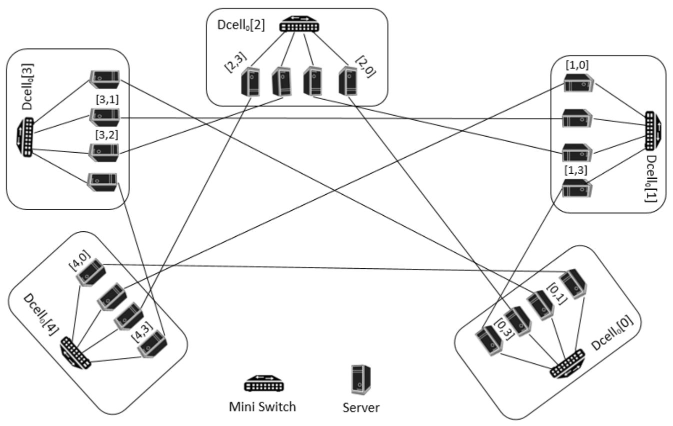


Fig. 2: The DCell topology.

3.3 Ficonn

Similar to DCell, every server uses the network interface card (NIC) with two types of ports, active port to connect to a mini-switch and a backup port for expansion. As shown in figure 3, a 0-level FiConn$_0$ contains an n-port switch and n servers. A FiConn$_S$ consists of $(\frac{p}{2}+1)$ FiConn$_{S-1}$'s, where p is the number of backup ports in a FiConn$_{S-1}$, and the number of servers is N$_S$=N$_{S-1}$ $(\frac{N_S-1}{2^p}+1)$, S≥1 [9].

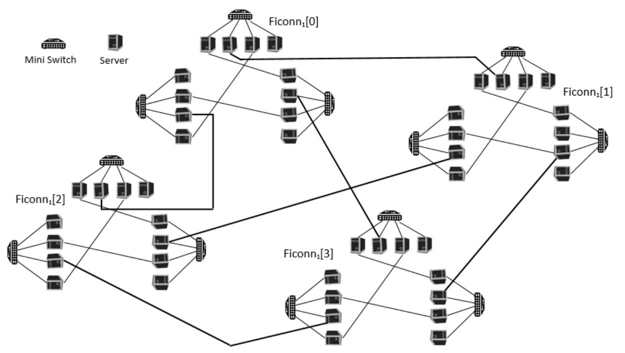


Fig. 3: The Ficonn topology.

4 Scheduling Problem

The scheduling problem is to determine plans for the operation of an industrial production system. In other words, it is about managing the allocation of resources to tasks over time, while satisfying at best a set of criteria and respecting certain constraints. The result of the process of solving a scheduling problem is a precise schedule of tasks to be performed which has three essential characteristics. Firstly, allocating the necessary resources to the tasks. Secondly, task sequencing and thirdly the date-marking that point out the start and end times of the tasks on the resources.

4.1 HSA Basics

Just like the optimization process to find an optimal overall solution of an objective function, the harmonization process relies on an orchestra playing a combination of harmonies to find the pleasant harmony determined by an aesthetic standard. To achieve this, a set of musicians proceeds by successive improvisations based on their experience. Each player sounds any pitch in the possible range, following one of these three rules firstly, play a tone of his memory, after that he plays a tone adjacent to the tone of his mind and finally, perform a random tone in the set of possible sounds[2]. HSA uses these three rules for generating a new solution. Here, the memory is a set of solutions generated randomly at the beginning. So the process produces at each iteration a new solution using either memory values, or modified values of the memory, or random values according to these parameters. Harmony Memory Considering Rate (HMCR), refers to control the choice of any value from Harmony Memory (HM). Pitch Adjusting Rate (PAR), which means a selection of a value adjacent to the amount of HM.

Standard HSA proceeds with the following steps: Start by initializing a set of parameters. Next step, improvise the HM to get a new harmony and then update it until some criteria are satisfied. We have added a randomization parameter to allow the selection of a random value in the range of possible values, to use it in a replacement that is applied in the evaluation step presented in **Section 5**, which allows the update of the memory to keep the best solutions and use them later.

4.2 Analogy Context

The job-shop scheduling type is known in the literature as a very complicated combinatorial problem and very difficult to solve. On account of its industrial origins, Table 1 gives a projection of necessary elements from an industrial workshop to a Cloud environment. A piece takes the place of an application process running inside a virtual machine (VM) which is software running within a server to imitate the behavior of a physical machine and have multiple Cloudlets that use the VM capacity and store its id to run a file on it.

Table 1: A workshop projection on Cloud environment.

Workshop	**Cloud Computing**
Piece	Application
Machine	Server(VM)
Operation (Task)	Cloudlet

From the analogy mentioned above, we used HSA to solve that context due to its ability to deal with the immensity of data that can be found in JSP, in addition to that, HSA provides a simple concept regarding its model and a smooth implementation besides, it relies on few adjustable parameters to speed up the convergence process.

5 Proposed Work

DCNs are at the heart of almost every private facility in Cloud computing, with a specific architecture pre-installed. This work takes into consideration the issue of application scheduling in the Cloud and adopting a JSP as a study case, that is suitable with the server-centric architecture, focusing on the DCell topology, to optimize the Makespan using HS process illustrated in figure 4.

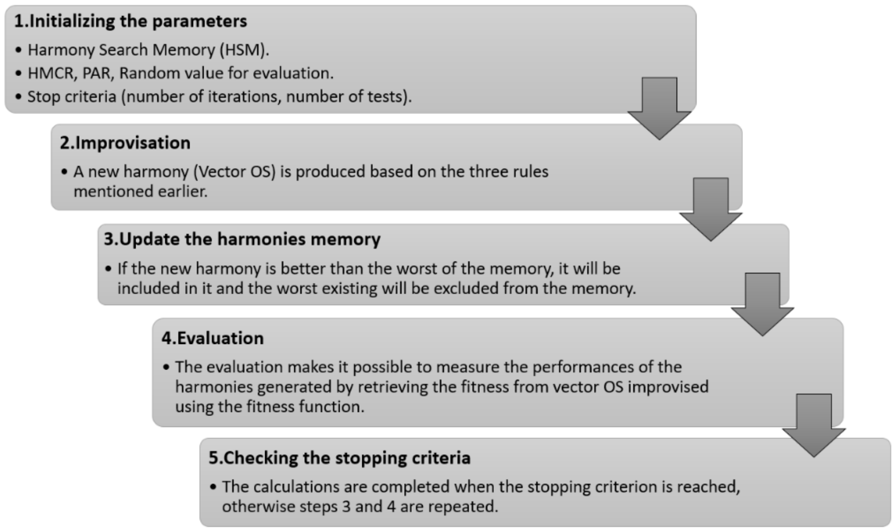

1.Initializing the parameters
• Harmony Search Memory (HSM).
• HMCR, PAR, Random value for evaluation.
• Stop criteria (number of iterations, number of tests).
2.Improvisation
• A new harmony (Vector OS) is produced based on the three rules mentioned earlier.
3.Update the harmonies memory
• If the new harmony is better than the worst of the memory, it will be included in it and the worst existing will be excluded from the memory.
4.Evaluation
• The evaluation makes it possible to measure the performances of the harmonies generated by retrieving the fitness from vector OS improvised using the fitness function.
5.Checking the stopping criteria
• The calculations are completed when the stopping criterion is reached, otherwise steps 3 and 4 are repeated.

Fig. 4: HS optimization procedure.

Improvisation is the most critical phase in the meta-heuristic HS because it allows the generation of a new harmony. It is based on the adjustment operator efficiency which we designed differently compared to the basic algorithm in order to avoid the risk of having unrealistic harmonies, this second part from the HS procedure is based on three dependent parts. Firstly, we run through the benchmark file to create a random harmony of Cloudlets in a no-discount draw which will be in the end a vector of Cloudlets (i.e., an OS vector), then add this vector to vector of vectors, all this inside a loop. Part two consists of adjusting this last vector by using an adjustment method that randomly arranges the index of the first Cloudlet in the vector OS, so that its new position takes the one of the first box containing a null value. The last part of the improvisation will receive an adjusted vector of vectors as an input to collect the cloudlets in a specific position from all the vectors and add it to a new final OS vector. This vector will be considered as the ultimate scheduling harmony and to be evaluated for the best fitness in afterward.

5.1 Implementation

The current work has been implemented using the toolkit NetworkCloudSim [6], which is an extension from CloudSim. It gives the possibility to create different types of DCNs, especially the hierarchical topologies with three separate switches level which are an edge switch, aggregate switch, and root switch. Initially, Net-CloudSim has a test example class implemented that represent a small DCN to test the communication between the servers and the edge switches. For the solutions coding, we followed a model of coding very used in the literature which is based on the component Operation Sequence (OS), it can be represented by a vector of integers of equivalent size to the number of Cloudlets and allows to give an order of execution of all the Cloudlets to execute in the Cloud. For example in Table 2, the second box containing the value five means that the second Cloudlet to be performed will be the fifth Cloudlet of the sixth application.

Table 2: Proposed OS vector.

Cl_{54}	Cl_{56}	Cl_{46}	Cl_{71}	Cl_{66}	Cl_{64}	Cl_{55}	Cl_{45}	Cl_{44}	Cl_{65}
5	5	4	7	6	6	5	4	4	6

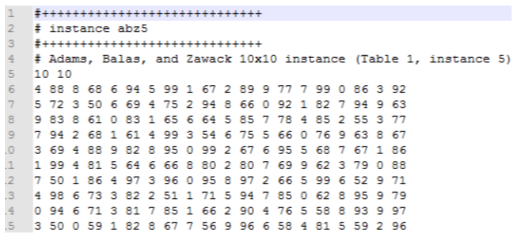
In our contribution, dealing with job-shop scheduling problem which is mono-objective, the objective function respects a single criterion which is the Makespan. The evaluation step from the HS procedure (**fig.4**) appends an extra data processing to the basic procedure and therefore a fitness function presented in **Algorithm 1** was developed. It proceeds by reading some VMs (N_VMs) from the benchmark file and then for each Cloudlet in the OS vector orderly, retrieve the affected machine and the corresponding duration from the problem data file, after that assign the cloudlet at the end of the current VM respecting the precedence constraints. After completing the scheduling, calculate the fitness of the solution and for that recovering the value of the Makespan.

Algorithm 1: Fitness calculation.

```
Input  : OS vector improvised (OSI).
Output: Fitness value.
Fitness = 0 ;
N = OSI size ;
read (N_VMs) ;
< $VM\_D$ > = Vector of VM duration ;
< $APP\_D$ > = Vector of application duration ;
for $K = 1$ to $N$ do
    Retrieve cloudlet index from the benchmark ($Cl\_Index$) ;
    Add ($Cl\_Index$) to the vector < $Cl\_Vec$ > ;
    Duration = Cloudlet run time in the first index of < $Cl\_Vec$ > ;
    Retrieve VM finish run time ($VMF\_RT$) ;
    Retrieve application finish run time ($APPF\_RT$) ;
    if ($VMF\_RT$) < ($APPF\_RT$) then
        (VMF_RT) = (APPF_RT) ;
        (VMF_RT) =(VMF_RT) + $Duration$ ;
        (APPF_RT) =(APPF_RT) + $Duration$ ;
    else
        (APPF_RT) =(VMF_RT) ;
        (APPF_RT) =(APPF_RT) + $Duration$ ;
        (VMF_RT) =(VMF_RT) + $Duration$ ;
    end
    < $VM\_D$ > = Set VM index with ($VMF\_RT$) value ;
    < $APP\_D$ > = Set an application index with ($APPF\_RT$) value ;
end
for $J = 1$ to $N\_VMs$ do
    M = J_index value from < $VM\_D$ > ;
    if $Fitness < M$ then
        $Fitness = M$ ;
    end
end
return Fitness ;
```

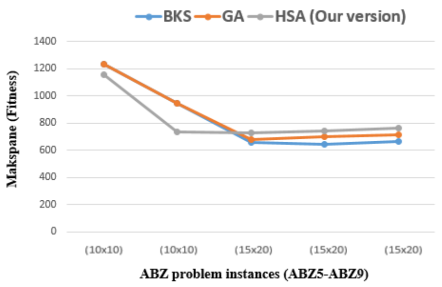
5.2 Experimentation and Results

In our study, we used benchmark samples ABZ, LA, FT, and ORB [13]. The file presented in figure 5a represent the Cloud user exigencies and can be read as follows: The fifth line represents the number of applications and the number of VMs in the Cloud (e.g., 10 applications and 10 VMs) respectively. The pair [4 88]) represents a single Cloudlet, such that the first digit (4) designates the id of the VM assigned to this Cloudlet and the second (88) indicates the execution time of the Cloudlet on this VM.



```
1    #++++++++++++++++++++++++++++++
2    # instance abz5
3    #++++++++++++++++++++++++++++++
4    # Adams, Balas, and Zawack 10x10 instance (Table 1, instance 5)
5    10 10
6    4 88 8 68 6 94 5 99 1 67 2 89 9 77 7 99 0 86 3 92
7    5 72 3 50 6 69 4 75 2 94 8 66 0 92 1 82 7 94 9 63
8    9 83 8 61 0 83 1 65 6 64 5 85 7 78 4 85 2 55 3 77
9    7 94 2 68 1 61 4 99 3 54 6 75 5 66 0 76 9 63 8 67
.0   3 69 4 88 9 82 8 95 0 99 2 67 6 95 5 68 7 67 1 86
.1   1 99 4 81 5 64 6 66 8 80 2 80 7 69 9 62 3 79 0 88
.2   7 50 1 86 4 97 3 96 0 95 8 97 2 66 5 99 6 52 9 71
.3   4 98 6 73 3 82 2 51 1 71 5 94 7 85 0 62 8 95 9 79
.4   0 94 6 71 3 81 7 85 1 66 2 90 4 76 5 58 8 93 9 97
.5   3 50 0 59 1 82 8 67 7 56 9 96 6 58 4 81 5 59 2 96
```

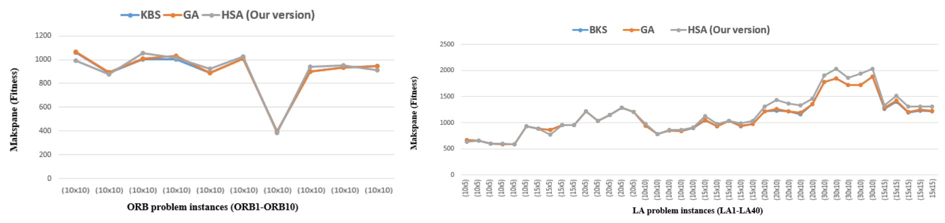
(a) ABZ5 benchmark(10x10)



(b) ABZ benchmark

Fig. 5: ABZ5 sample instance and ABZ comparison result

In order to determine the best set of parameters, several series of tests were performed on five instances of the ABZ class (ABZ5, ABZ6, ABZ7, ABZ8, ABZ59), five times each by combining different parameter values of HMCR and PAR with 300000 iterations. In what follows an explanation in detail on the set of tests carried out. The experiments were performed on an Intel i5-6200U CPU up to 2.8GHz with 4gb of Ram. Initially, a size of 100 of HM is more than enough according to the calculated fitness. The best combination of HMCR and PAR parameters is decided after fixing the PAR each time to a given value belonging to the following set of values: (0.1; 0.2; 0.01) and by varying the HMCR between (0.7; 0.8; 0.9). The best combination obtained is PAR = 0.01 and HMCR = 0.9. Final tests are required to validate our implementation results, by comparing our best-obtained fitness from ABZ, LA, ORB instances to the knowing best solution (KBS) in the literature and also to Genetic Algorithm (GA).



(a) ORB benchmark (b) LA benchmark

Fig. 6: ORB and LA comparison result

From the previous figures, we notice that our method gives on the majority of the benchmarks the same fitness values found in the literature, and manages to exceed them, as in figure 6a with 40% of ORB instances that our HS approach Makespan was better than what was found. Same goes in figure 6b with 32.5% of LA instances and 40% of ABZ instances in figure 5b.

6 Conclusion

Despite the constant evolution of Cloud Computing, The optimal resolution of Job Shop type scheduling problems is in some cases impossible, owing that to their size and complexity of the problem are taken into account when choosing the optimization method. In summary, we have presented an approach based on HSA to schedule a set of applications with a DCell topology in a Cloud environment using NetCloudSim. The results obtained are mostly satisfactory and remain in their encouraging overall, compared to those found in the literature. In perspective, we plan to integrate meta-heuristics for local search to improve resources utilization and services performance in the Cloud and explore other types of scheduling issues such as flexible job-shop, flow-shop, and hybrid flow-shop.

References

1. Chaudhary, N., Kalra, M.: An improved Harmony Search algorithm with Group technology model for scheduling workflows in cloud environment. In: 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON), ISBN 978-1-5386-3004-4, IEEE, Mathura (2017).
2. Zong, W. G., Joong, H. Kim., Loganathan, G. V.: A New Heuristic Optimization Algorithm: Harmony Search, SIMULATION: Transactions of The Society for Modeling and Simulation International (SIMUL-T SOC MOD SIM),Vol. 76, pp.60-68 (2001).
3. Fathi, M.H., Khanli, L.M.: Consolidating VMs in Green Cloud Computing Using Harmony Search Algorithm. In: Proceedings of the 2018 International Conference on Internet and e-Business, pp. 146-151.ACM, Singapore (2018).
4. Al- maamari, A., Omara, F.A.: Task Scheduling using Hybrid Algorithm in Cloud Computing Environments. IOSR Journal of Computer Engineering 17(3), 96-106 (2015).
5. Calheiros, R., Ranjan, R., Beloglazov, A., De Rose, C., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software-Practice and Experience, vol. 41, 23-50 (2011).
6. Saurabh, K.G., Rajkumar, B.: NetworkCloudSim: Modelling Parallel Applications inCloud Simulations, IEEE, International Conference on Utility and Cloud computing, vol.76, pp. 105-113. Victoria (2011).
7. Melnik, M., Trofimenko, T.: Polyrhythmic Harmony Search for Workflow Scheduling. In: 4th International Young Scientists Conference on Computational Science, vol. 66, pp. 468-476. Procedia Computer Science, Athens (2015).
8. Haoqian. M, Lianglun. Ch.: ACO-HS: A Hybrid Cloud Resource Scheduling Model, 3rd International Conference on Engineering Technology and Application, ISBN 978-1-60595-383-0, Kyoto (2016).
9. Tao, C., Xiaofeng, G., Guihai, C.: The features, hardware, and architectures of data center networks: A survey. Journal of Parallel and Distributed computing, 45-74 (2016).
10. Guo, C., Lu. G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., Lu, S.: Bcube: a high performance, server-centric network architecture for modular data centers, SIGCOMM Comput. Commun. Rev. 39 (4) 63–74, (2009).
11. Guo, C., Wu, K., Tan, L., Shi, Y., Tian, C., Zhang, Y., Lu, S.: DCell: a scalable and fault-tolerant network structure for data centers, ACM SIGCOMM Comput. Commun. Rev. 38 (4) 75–86, (2008).
12. Li, D., Guo, H., Tan, Y., Zhang, Y., Lu, S.: FiConn: Using backup port for server interconnection in data centers, in: IEEE INFOCOM, pp. 2276–2285. (2009).
13. Github, https://github.com/tamy0612/JSPLIB, last accessed 2018/12/09.
14. Hammadi, A., Mhamdi , L.: A survey on architectures and energy efficiency in Data Center Networks, Vol. 40, 1-21 (2014).
15. TING, W., ZHIYANG, S., YU X., MOUNIR, H.: Rethinking the Data Center Networking: Architecture, Network Protocols, and Resource Sharing. IEEE Access, Vol.2, 1481 – 1496 (2014).