International Journal of Grid Distribution Computing
Vol. 8, No.5, (2015), pp.245-256
http://dx.doi.org/10.14257/ijgdc.2015.8.5.24

Task Scheduling Using PSO Algorithm in Cloud Computing Environments

[1]Ali Al-maamari and [2]Fatma A. Omara

[1]Department of Computer Science, Cairo University, Egypt
Alialmamri2011@gmail.com
[2]Professor, Department of Computer Science, Faculty of Computers & Information,
Cairo University, Egypt

Abstract

The Cloud computing has become the fast spread in the field of computing, research and industry in the last few years. As part of the service offered, there are new possibilities to build applications and provide various services to the end user by virtualization through the internet. Task scheduling is the most significant matter in the cloud computing because the user has to pay for resource using on the basis of time, which acts to distribute the load evenly among the system resources by maximizing utilization and reducing task execution Time. Many heuristic algorithms have been existed to resolve the task scheduling problem such as a Particle Swarm Optimization algorithm (PSO), Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Cuckoo search (CS) algorithms, etc. In this paper, a Dynamic Adaptive Particle Swarm Optimization algorithm (DAPSO) has been implemented to enhance the performance of the basic PSO algorithm to optimize the task runtime by minimizing the makespan of a particular task set, and in the same time, maximizing resource utilization. Also, .a task scheduling algorithm has been proposed to schedule the independent task over the Cloud Computing. The proposed algorithm is considered an amalgamation of the Dynamic PSO (DAPSO) algorithm and the Cuckoo search (CS) algorithm; called MDAPSO. According to the experimental results, it is found that MDAPSO and DAPSO algorithms outperform the original PSO algorithm. Also, a comparative study has been done to evaluate the performance of the proposed MDAPSO with respect to the original PSO.

Keywords*: Cloud computing, Particle Swarm Optimization, Task scheduling*

1. Introduction

Cloud computing is a new computing technology to enhance the virtualized resources designed for end users in a dynamic environment in order to provide reliable and trusted service [1]. The offered services includes but not limited to the possibility of building application & other various services through internet virtualization which is the main technique which improve physical resources utilization in cloud computing [2]. It allows abstraction and isolation of underlying physical resources and reduce required hardware equipment. The technique – virtualization – include network virtualization end enable the operator to create virtual machines on a single physical server. Specifically, virtual machines can be designed by increasing or decreasing CPU power & number of CPUs, in order to efficiently utilize the virtualized resources that execute the computing intense application [3].

Establish an efficient load balanced algorithm would be proposed to ensure cloud computing could be used efficiently & effectively which is a fundamental goal of the service providers. There is a real need to develop new task scheduling algorithm to meet the virtualization principles & demand. The objective of the task scheduling algorithm is

ISSN: 2005-4262 IJGDC

to achieve high system throughput, improve the load balance and reduce the completion period while ensuing in the same time meeting the job requirement with available virtualized resources. According to the task scheduling, a set of an appropriate number of tasks is to be scheduled to the virtual machines. Tasks scheduling over the Cloud Computing resources are the most important task because the user will have to pay for resource use on the basis the time.

Many meta-heuristic algorithms have been proposed such as PSO algorithm that is appropriate for dynamic task scheduling include. On the other hand, Particle Swarm Optimization (PSO) has become popular because of its simplicity and its effectiveness in a broad range of application. Some of the applications that have used PSO to solve NP-Hard problems like Scheduling problem [4], and the task allocation problem [5].

Kennedy and Eberhart [6] have presented a self-adaptive global search based optimization technique. The algorithm is similar to other population-based algorithms as Genetic algorithms (GA) without direct recombination of individuals of the population. Instead, it relies on the social behavior of the particles. In every generation, all particles adjust its trajectory based on its best position and the position of the best particle (global best) of the entire population. These concepts increase the stochastic nature of the particle and converge quickly to a global reduce with a reasonable right solution. Xin, G. Chen [7] has proposed a dynamic adaptive particle swarm optimization. The target of this algorithm is improve the PSO algorithm affinity problem in inertia weight where great inertia weight facilitates a global search while a little inertia weight facilitates a local optimal of the High dimensional function optimization problems to improve its global optimal and Affinity speed. A dynamic adaptive strategy was offered into the variant to edit the inertia weight value according to the current swarm variety and collection degree, as well as, the effect on the search performance of the swarm.

According to the work in this paper, an algorithm based on particle swarm optimization algorithm called Dynamic Adaptive Particle Swarm Optimization (DAPSO) algorithm has been proposed. Also, another algorithm has been proposed by amalgamation the proposed DAPSO with a cuckoo search algorithm, called (MDAPSO), to optimize the task scheduling in the Cloud environment to minimize the completion time and increase the resource utilization.

The rest of this paper is organized as follows. Section 2 summarizes previous related work in this field. Section 3 describes the proposed system, model. Sections 4, 5 and 6 describe the intelligent PSO, DAPSO, and MDAPSO algorithms, and Sect. 7 describes proposed algorithm in detail. Section 8 compares our proposed algorithm with several similar algorithms, and the final section presents our conclusion.

2. Related Work

Task scheduling has been a significant research topic whose objective is to ensure that every computing resource is distributed efficiently and equitably and at the end improves resource utility. In traditional computing environments of distributed computing, grid and parallel computing, a set of scheduling strategies have been proposed.

R. P. BRENT [8] has proposed a first-fit strategy, and it is used by some cloud system such as Eucalyptus in Nurmi, D. Wolski *et al.,* [9] In these methods, the starvation problem is nearly solved, and the makespan is reduced. But, the requests will execute on each resource and would not support the optimal usage of resources and appropriate load balancing.

Wang, X., Yeo *et al.,* [10] has proposed the Look-Ahead Genetic Algorithm (LAGA) for large-scale distributed systems such as Grid and Cloud, based on GA algorithm. It is a computation-intensive and reliability driven reputation algorithm that considers the tasks runtime using the task failure rate (task failures each unit time) of resources in order to define the status and evaluate the reliability of resources. This algorithm computes a task

ordering procedure by the resource completion time in all generation and choice a resource with the least failure rate in mutation operation. It focuses on completion time and schedule failure rate.

Singh, J., Singh, H. [11] has proposed the Node duplication Genetic (NGA) algorithm which has been used for heterogeneous multiprocessor systems, based on GA method. The algorithm concerns about the completion time of application and communication delay time. The fitness function in NGA algorithm does the evolution in two different stages. The first stage is the fitness of task that the authors said it equipped the system with the knowledge of all the tasks are executed and scheduled in a legal order. Here the legal system is scheduled a pair of tasks on a single processor while the pair of tasks is independent of each other. The second stage is the fitness of processor that attempts to minimize the processing time.

Pooranian *et al.*, [12] have proposed a task scheduling technique for grid computing based on a merge PSO with the gravitational emulation local search (GELS) algorithm that minimizes makespan and the number of tasks that fail to meet their deadlines.

Lee [13], has proposed a dynamic load balancing algorithm on the basis of an existing algorithm called WLS (weighted least connection). According to this algorithm, the tasks are assigned to node according to the number of connections that exist for this node. It is comparing a set of connections from each node in the cloud and the task assigned to the node with the lowest number of connections. Nonetheless, weighted least connection would not take into account the capabilities of each node, such as processing speed, storage, capacity, and bandwidth.

Ren [14], has improved the algorithm of the WLC [13], with taking into account the time series and trials. This modified algorithm called Exponential Smooth Forecast based on Weighted Least Connection (ESWLC).The ESWLC steps are; 1) build the conclusion of the assignment of a particular task to the node after having a number of tasks assigned to that node and identify node capabilities. 2) Build a decision on the basis of the experience of the node's memory, CUP memory, the number of connections and the amount of disk space currently being used. 3) Then, predict which node is to be chosen on the basis on exponential smoothing.

L. Zhang, *et al*., [15] has proved that the particle swarm optimization algorithm can get the better schedule than the genetic algorithm in grid computing. A. Salman [16] has illustrated that the performance of Particle Swarm Optimization (PSO) algorithm is better than GA algorithm in distributed system. Not only is the PSO algorithm solution quality better than GA in a majority of the test cases, but also PSO algorithm run quicker than GA.

Xin Lu, Zilong Gu. [17] Dong, Wang, D. [18] and Song, X., L.[19] have proposed a Load balancing task scheduler to balance the fully system load while trying to minimize the makespan of a specific task set. They used two different load balancing scheduling algorithms based on the solution of ant colony optimization (ACO) technique, which aims to minimize the completion time based on pheromone.

A. Al-maamari, F. Omara.[20] have proposed a task scheduling algorithm for cloud computing based on a merge PSO algorithm with the Cuckoo Search (CS) algorithm, called (PSOCS), the task assigned to the virtual machine, that aims to minimizes makespan and the maximum resource utilization .

According to the work in a paper, the Particle Swarm Optimization has been concerned to optimize the task scheduling problem with focusing on minimizing the total executing time.

3. The Scheduling System

Figure 1 illustrates an overview of the scheduling system. the system consists of three modules, the first module is the application which represents the set of the cloudlets (tasks). The second module is the Mapping Algorithms (MA) which estimates the

expected time for each cloudlet to allocate on each virtual machine, and it is assumed that these values are available to the scheduler. A third module is a virtual machine (VMs) which used to execute the cloudlets. The cloudlets expected time have been stored in an m × n matrix, where m is the number of virtual machines, and n is the number of cloudlets. Obviously, n/m will generally be greater than 1, with more cloudlets than virtual machines, so that some machines will need to be assigned multiple cloudlets. The Estimated Running Time (ERT) is defined as the time of executing task j on resource r [21]. Each column i of the expected running times (ERT) matrix contains the expected running time (ERT) of each cloudlet j on machine i.

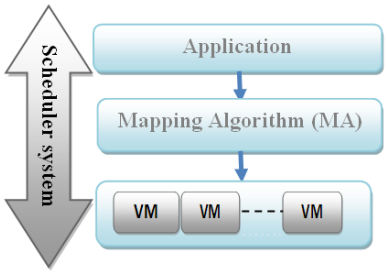

Figure 1. Scheduling System

The main objective of allocating tasks on virtual machines is to reduce the makespan. The makespan of a task is defined as the overall task completion time. We denote completion time of task T_i on VM_j as CT_{ij}. Hence, the makespan is defined using the following equation [22]:

$$Makespan = \mathrm{CT_{max}}[i, j] \mid i \epsilon T,\ i = 1,2, \ldots n \text{ and } j \epsilon VM, j = 1,2,\ldots m \qquad (1)$$

Where $\mathrm{CT_{max}}[i, j]$ is the maximum completion time of task $_i$ on a VM $_j$., and n, m are the number of tasks and virtual machines respectively.

Let $VM = VM_1, VM_2, \ldots . VM_m$ be the number of m virtual machines that must be processed n tasks represented by the group $T = T_1, T_2, \ldots T_n$. The virtual machines are parallel and independent, and the schedule allocates independent tasks to these VM_s. Also, the Processing a task on a virtual machine cannot interrupt (i.e.) Non-preemption. We denote end time of a task T_i by CT_{ij}. The aim of the proposed algorithms is to reduce the $Makespan$ which can be denoted as CT_{max}. The run time of each task for each virtual machine must be calculated for the purpose of scheduling, If the processing speed of a virtual machine VM_j is PS_j, then the processing time for task P_i can be calculated by equation. (2) [23]:

$$Tij = Ci / PSj \qquad (2)$$

Where P_{ij} is the processing time of task P_i by virtual machine VM_j and C_I is the computational complexity of the task P_i [23]. The processing time P_{ij} of each task P_i on VM_j are stored in the runtime matrix. The processing time of each task in the virtual machines can be calculated by equation (3):

$$P_j = \sum_{i=1}^{n} Pij \qquad (3)$$

According to (1), (2) and (3), the task scheduling algorithm should satisfy the following equation

$$\sum_{i=1}^{n} Pij \ \leq CT_{max} \qquad (4)$$

By considering the load balancing, the tasks will be transferred from one VM to other to reduce CT_{max}, as well as, response time. The processing time of a task varies from one

VM to another based on the speed of the virtual machines. In case of transferring, the completion time of a task may vary because of load balancing, optimally.

The main objective of the proposed task scheduling MDAPSO algorithm is that the tasks should be allocated on the virtual machine in order to minimize the makespan and maximize the resource utilization. Therefore, the proposed DAPSO algorithm has been introduced to enhance the performance of the original PSO algorithm.

4. The Basic Particle Swarm Optimization (Pso) Algorithm

The PSO algorithm has been introduced by Kennedy and Eberhartin 1995 [6], [24]. According to the PSO algorithm, the particle set of a process can be presented according to the following equations.

$$V_{id} = V_{id} + c_1 rand_1 (P_{id} - X_{id}) + c_2 rand_2 (P_{gd} - X_{id}) \quad (5)$$

$$X_{id} = X_{id} + V_{id} \quad (6)$$

Where Vid and Xid are the velocity of particle i at iteration d, $rand_1$ and $rand_2$ are random numbers between 0 and 1, $v_{id} = [-v_{max}, v_{max}]$ where V^{max} is constant, c1 and c2 are learning factors called the cognition and the social parameters. The ith particles in a D-dimensional vector is $X_i = (X_{i1}, X_{i2}, \ldots, X)$ in the space of flight speed is *Vi* and $V_i = (V_{i1}, V_{i2}, \ldots, V_i)$. The ith is known so far to search the optimum position is $P_i = (P_{i1}, P_{i2}, \ldots, P_i)$, and the particle swarm so far to search the optimal position is $P_g = (P_{g1}, P_{g2}, \ldots, P_g)$. In [25], the speed equation (5) has submitted the following changes:

$$V_{id} = w\, V_{id} + c_1 r_1 (P_{id} - X_{id}) + c_2 r_2 (P_{gd} - X_{id}) \quad (7)$$

Where W is inertia weight. Equations (5) and (7) are considered the basic of PSO iterative formula. For more details about w value see [24], also in section 5.

Resolving the task scheduling problem using PSO algorithms that how to enter a schedule as a search solution, find appropriate maps among problem solutions and PSO particles. According to the proposed DAPSO algorithm, each particle represents a possible solution for the task assigned using an array of n elements, where all elements randomly produce integer values between 1 and m. Figure 2 shows the assignment of ten tasks to five virtual machines. For Example, in Particle 1 or nest 1, tasks T_1, T_5 and T_3 are assigned to VM_1 and tasks T_2, T_6 and T_4 are assigned to VM_2 and T_7 is assigned to VM_3 and T_8 and T_9 are assigned to VM_4, and T_{10} is assigned to VM_5.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
Particle1	VM1	VM2	VM1	VM2	VM1	VM2	VM3	VM4	VM4	VM5
Particle2	VM2	VM2	VM3	VM2	VM4	VM1	VM1	VM5	VM3	VM5
Particle3	VM3	VM3	VM2	VM3	VM5	VM4	VM1	VM4	VM2	VM4
	…….	…….								

Figure 2. Particle Representation

5. Dynamic Adaptive Particle Swarm Optimization

The objective of the proposed DAPSO is to solve the PSO affinity problem in inertia weight where great inertia weight facilitates a global search while a little inertia weight facilitates a local search. The inertia weight formula that was used is represented in Eq (8) [26]:

$$W_t = W_{min} + (W_{max} - W_{min})F_t\varphi_t \quad (8)$$

Where W_{min} and W_{max} are the minimum and maximum inertia weight values, t is the current number of iterations, the diversity function F*t* and adjustment function φ_t , both in the ith iteration, are represented in equations (9) and (10), respectively:

$$F_t = 1 - \left(\frac{2}{\pi}\right) \arctan(E) \quad (9)$$

Where E is the group fitness as shown in equation (11):

$$\varphi_t = e^{(-t2/(2\cdot2))} \quad (10)$$

Where $\sigma = T/3$ and T are the total numbers of iterations:

$$E = 1 - \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - f_{avg})^2 \quad (11)$$

Where N is the swarm size, $f(x_i)$ is the fitness of particle i, and f_{avg} is the current average fitness of the swarm, and it is represented in equation (12):

$$f_{avg} = \frac{1}{n}\sum_{i=1}^{N} f(x_i) \quad (12)$$

The proposed dynamic adaptive particle swarm optimization (DAPSO) algorithm can be summarized as follows:

1. Initialize the particle swarm randomly.
2. Evaluate the fitness function for each particle.
3. Compare the optimization fitness value of each particle with its previous best position pbest, If the pbest is better than the previous pbest values, then set the current pbest value as the best value of the particle.
4. Compare the fitness function value of each particle with the previous gbest value, if the current value is better than the previous gbest values, then reset the current value as the global best value of the swarm.
5. To all particles of particle swarm, execute the following operations:
   - Update the position and velocity of particles by using equations (6) and (7).
   - Update the inertia weight by using equations (11) (12) (8) and (10).
6. Check if a stop criterion is met. I fit is met, the execution is terminated. Otherwise, go to Step 2.

6. Cuckoo Search Algorithm

The CS is an optimization algorithm developed by Xin-she Yang and Suash Deb in 2009 [27]. It was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds (of other species). Some host birds can engage direct conflict with the intruding cuckoos. For example, if a host bird discovers the eggs are not their own, it will either throw these alien eggs away or simply abandon its nest and build a new nest elsewhere. According to the CS algorithm, when generating new solutions $X^{(t+1)}$ for a cuckoo i, a Lévy flight is performed using the equation (13) [27].

$$X^{(t+1)} = X^{(t)} + \alpha \wedge Lévy(\lambda) \quad (13)$$

Where $X^{(t)}$ represents the current location, $\alpha > 0$ is the step size, which should be related to the scales of the problem that the algorithm is trying to solve. In most cases, $\alpha = 1$, and $\lambda \in (0, 3)$ are used. Equation (13) is in core stochastic equation for a random walk which is similarly of a Markov chain who's next location (status) depends on two parameters; current location (the first term in equation. 13) and the possibility of

transmission (the second term in Eq. 13).The product ^ represents entry-wise multiplication [28].

Cuckoo characteristics could be described, as a model for good behavior other animals have extensive use in computing Intelligence Systems [29]. According to the CS algorithm, an initial set of nests, which represent the solutions, are randomly generated. These solutions are then updated over multiple generations. The process of updating an individual solution is as follows; a random nest is chosen, and a new solution is generated by random walking from this previous solution. This new solution can then replace a different randomly chosen solution if it has a fitness value better than the original. After this possible replacement of a solution, all of the nests are ranked by fitness and the worst fraction of the nests is replaced with random solutions. This combination of mechanisms allows the solutions to search locally and globally at the same time for the optimal solution [28].

7. The Proposed Mdapsotask Scheduling Algorithm

The PSO algorithm uses different search stages, and these stages are close to the optimum stage with their pbest and gbest values. PSO can be used for continuous and discrete problems, and it is good for global searches in the problem space. But it is weak for local searches, with a large possibility of becoming trapped in a local optimum in the last iteration. PSO converges globally because it searches globally. It always tries to move to solutions that have better fitness functions in a purely stochastic search problem space. It does not pay close attention to local subspaces so it is unable to recognize and avoid local optima. As a result, PSO may become trapped in local optima and have a low convergence rate in the late iterative process. There is option for addressing this problem is to use local search algorithms such as CS algorithm that can avoid local optima.

According to the proposed MDAPSO task scheduling algorithm, the PSO algorithm has been used as the main search algorithm, while the DAPSO and CS algorithm are used to improve the population. There are two reasons for using both algorithms. First, we needs an algorithm that based on a population to search the entire cloud space for this problem. Second, the cloud environment is dynamic. So, the scheduling algorithm must be fast enough to adapt with the natural cloud environment and must be able to converge faster than other algorithms. Moreover, DAPSO has been introduced to overcome the weakness of the original PSO algorithm for local searches because the inertia weight technique is very useful to ensure convergence. However, there is a disadvantage of DAPSO algorithm is that once the inertia weight is decreased, it cannot increase if the swarm needs to search new areas. This algorithm not able to recover its exploration mode. Therefore, by combining the DAPSO algorithm with CS algorithm is considered powerful in searches addresses this problem because CS has a set of mechanisms allows the solutions to search locally and globally at the same time for the optimal solution. Generally, the DAPSO algorithm is used to solve the PSO convergence problem in inertia weight where great inertia weight facilitates a global search while a little inertia weight facilitates a local search, also control the balance between global and local exploration, to obtain quick convergence, and to reach an optimum. CS algorithm is run on the global result of the last iteration of the DAPSO. That is an initial solution for CS which is provided by DAPSO during the mix search process.

8. The Performance Evaluation

8.1. Experimental Settings

Cloudsim3.0.3 is an open source simulator which has been developed by Gridbus project team and the grid Laboratory of the University of Melbourne in Australia. The Cloudsim can run on Linux and Windows systems [30]. CloudSim has been used to implement the proposed MDAPSO task scheduling algorithm. Also, a comparative study

has been done to evaluate the performance of the proposed DAPSO, and MDAPSO algorithms with respect to the PSOCS algorithm [20], and the original PSO algorithm. This simulation mainly validates the advantage of the makespan and the resource utilization among these scheduling algorithms in the Cloud Computing environment.

8.2. Performance Results

To evaluate the performance of the four algorithms; original PSO, DAPSO, PSOCS and MDAPSO, 5 five and ten Virtual machines are considered with 10, 20, 30, and 40 cloudlets.

Tables 1 represent the makespan of PSO, DAPSO, PSOCS and MDAPSOalgorithms using five virtual machines and a set of various cloudlets respectively.

Table 1. Compared Scheduling Algorithm With Execution Time (Sec)

PSO	DAPSO	PSOCS	MDAPSO	VM	Cloudlet
3.200596	2.619682	2.254691	2.154	5	10
5.627895	4.514721	3.59614	3.475		20
8.896552	8.246502	6.436863	4.691		30
13.31111	10.25296	9.242986	8.48		40

According to the results in Figure 3, the proposed MDAPSO algorithm, with the respect to the execution time using 5 VMs, outperforms the default PSO, DAPSO and PSOCS algorithms by 38.63% and 25.30% respectively.

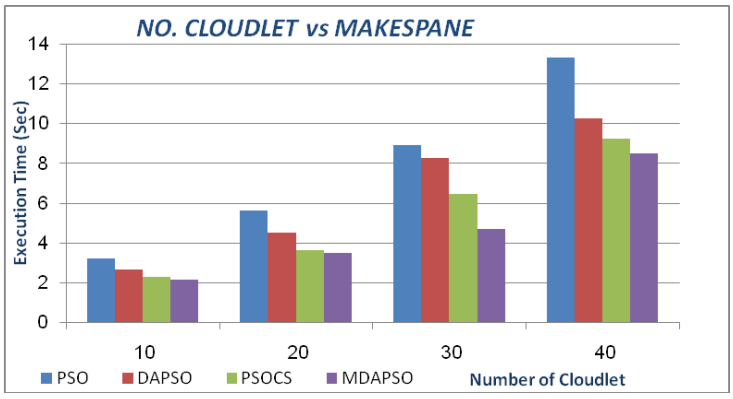

Figure 3. The ExecutionTime when No. VMs (5)

Tables 2. represent the makespan of PSO, DAPSO, PSOCS and MDAPSO algorithms using ten virtual machines and a set of various cloudlets respectively.

Table 2. Compared Scheduling Algorithm with Execution Time (Sec)

PSO	DAPSO	PSOCS	MDAPSO	VM	Cloudlet
2.259	1.7412	1.4830	1.372	10	10
3.893	2.9823	2.8084	2.093		20
7.235	5.7235	4.2235	3.981		30

8.595	5.2986	4.8917	4.581		**40**

According to the results in Figure 4, the proposed MDAPSO algorithm, with the respect to the execution time using 10 VMs, outperforms the default PSO, DAPSO and PSOCS algorithms by 44.29% and 23.75% respectively.

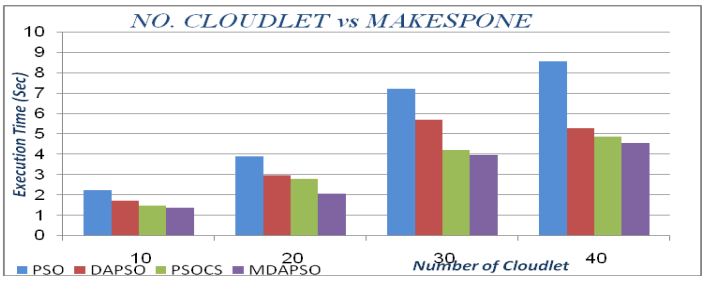

Figure 4. The Execution Time Of All Cloudlet when No. VMs (10)

The simulation results of the resource utilization of PSO, DAPSO, PSOCS and MDAPSO algorithms using five virtual machines and a set of various cloudlets respectively are described in Table 3. and Figure 5.

Table 3. Compared Scheduling Algorithm With Utilization

PSO	DAPSO	PSOCS	MDAPSO	VM	Cloudlet
0.626073	0.733601	0.854172	0.8672	5	**10**
0.594803	0.718096	0.919375	0.954354		**20**
0.616211	0.681181	0.92038	0.972168		**30**
0.52374	0.57813	0.843596	0.961894		**40**

According to the results in Figure 5, the proposed MDAPSO algorithm, with the respect to the resource utilization using 5 VMs, outperforms the default PSO, DAPSO and PSOCS algorithms by 36.19% and 20.60% respectively.

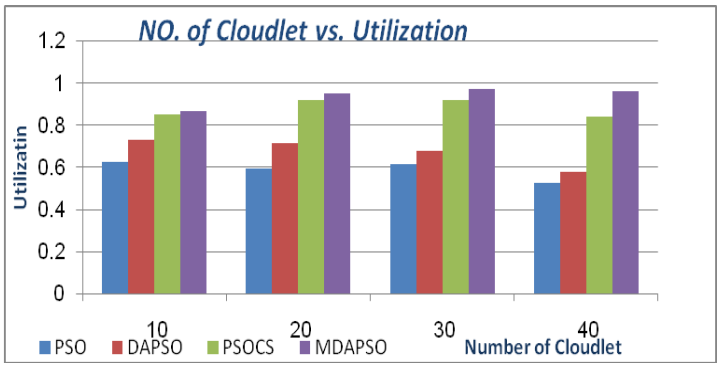

Figure 5. Comparison Utilization Of Number Of Cloudlets

Table 4. represent the resource utilization of PSO, DAPSO, PSOCS and MDAPSO algorithms using 10 virtual machines and a set of various cloudlets respectively.

Table 4. Compared Scheduling Algorithm With Resource Utilization

PSO	DAPSO	PSOCS	MDAPSO	VM	Cloudlet
0.42966	0.60066	0.65233	0.7012	10	10
0.43191	0.62932	0.69743	0.7933		20
0.32182	0.45051	0.56571	0.7572		30
0.39871	0.71601	0.80701	0.8909		40

According to the results in Figure 6 the proposed MDAPSO algorithm, with the respect to the resource utilization using 10 VMs, outperforms the default PSO, DAPSO and PSOCS algorithms by 49.26% and 23.79% respectively.

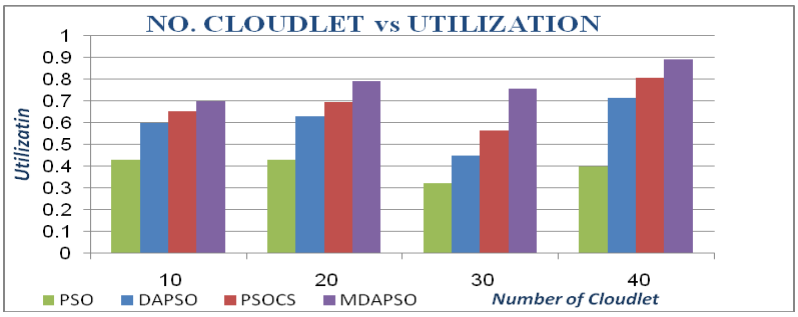

Figure 6. Comparison Utilization of Number of Cloudlets

9. Conclusion

According to the work in this paper, a dynamic adaptive particle swarm optimization (DAPSO) has been introduced and implemented to solve the PSO affinity problem in inertia weight where great inertia weight facilitates a global search while a little inertia weight facilitates a local search. Also, a new task scheduling algorithm has been introduced to minimize the makespan and increase the utilization ratio of application workflows on the Cloud computing. This new algorithm is considered an amalgamation of the DAPSO and CS algorithms, called MDAPSO algorithm, where DAPSO algorithm is used to improve the inertia weight and CS algorithm is used in the local search where the performance is improved by changing inertia weight and trapping on local search has been improved. To evaluate the proposed MDAPSO task scheduling algorithm, a comparative study among the proposed MDAPSO, original PSO, DAPSO, and PSOCS algorithms has been done. According to the experimental results, it is found that the proposed MDAPSO algorithm outperforms the original PSO, DAPSO, and PSOCS algorithms with respect to the makspam and resource utilization. In addition, the MDAPSO and DAPSO algorithms perform better performance than the original PSO algorithm.

References

[1] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges", in Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, **(2010)**, pp. 27-33.
[2] F. Baroncelli, B. Martini and P. Castoldi, "Network virtualization for cloud computing", annals of telecommunications-annales des télécommunications, vol. 65, **(2010)**, pp. 713-721.
[3] D. Hensgen and R. F. Freund, "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems", Journal of Distributed Computing, Special Issue on software support for distributed computing, vol. 59, **(1999)**.
[4] B. Yu, X. Yuan and J. Wang, "Short-term hydro-thermal scheduling using particle swarm optimization method", Energy Conversion and Management, vol. 48, **(2007)**, pp. 1902-1908.

[5] P. -Y. Yin, S. -S. Yu, P. -P. Wang and Y. -T. Wang, "A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems", Computer Standards & Interfaces, vol. 28, **(2006)**, pp. 441-450.
[6] J. Kennedy and R. Eberhart, "Particle swarm optimization", in Neural Networks, 1995. Proceedings, IEEE International Conference on, vol. 4, **(1995)**, pp. 1942-1948.
[7] J. Xin, G. Chen and Y. Hai, "A particle swarm optimizer with multi-stage linearly-decreasing inertia weight", in Computational Sciences and Optimization, 2009 CSO 2009, International Joint Conference on, **(2009)**, pp. 505-508.
[8] R. P. Brent, "Efficient implementation of the first-fit strategy for dynamic storage allocation", ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 11, **(1989)**, pp. 388-403.
[9] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, *et al*., "The eucalyptus open-source cloud-computing system", in Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on, **(2009)**, pp. 124-131.
[10] X. Wang, C. S. Yeo, R. Buyya and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm", Future Generation Computer Systems, vol. 27, **(2011)**, pp. 1124-113.
[11] J. Singh and H. Singh, "Efficient tasks scheduling for heterogeneous multiprocessor using genetic algorithm with node duplication", Indian J. Comput. Sci. Eng, vol. 2, **(2011)**, pp. 402.
[12] Z. Pooranian, M. Shojafar, J. H. Abawajy and A. Abraham, "An efficient meta-heuristic algorithm for grid computing", Journal of Combinatorial Optimization, **(2013)**, pp. 1-22.
[13] R. Lee and B. Jeng, "Load-balancing tactics in cloud", in Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on, **(2011)**, pp. 447-454.
[14] X. Ren, R. Lin and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast", in Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on, **(2011)**, pp. 220-224.
[15] L. Zhang, Y. Chen, R. Sun, S. Jing, and B. Yang, "A task scheduling algorithm based on PSO for grid computing", International Journal of Computational Intelligence Research, vol. 4, **(2008)**, pp. 37-43.
[16] A. Salman, I. Ahmad and S. Al-Madani, "Particle swarm optimization for task assignment problem", Microprocessors and Microsystems, vol. 26, **(2002)**, pp. 363-371.
[17] X. Lu and Z. Gu, "A load-adapative cloud resource scheduling model based on ant colony algorithm", in Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on, **(2011)**, pp. 296-300.
[18] K. Li, G. Xu, G. Zhao, Y. Dong and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization", in Chinagrid Conference (ChinaGrid), 2011 Sixth Annual, **(2011)**, pp. 3-9.
[19] X. Song, L. Gao and J. Wang, "Job scheduling based on ant colony optimization in cloud computing", in Computer Science and Service System (CSSS), 2011 International Conference on, **(2011)**, pp. 3309-3312.
[20] A. Al-maamari and F. Omara, "Task Scheduling using Hybrid Algorithm in Cloud Computing Environments", IOSR Journal of Computer Engineering (IOSR-JCE), vol. 17, **(2015)**, pp. 96-106.
[21] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, *et al*., "Task scheduling strategies for workflow-based applications in grids", in Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on, **(2005)**, pp. 759-767.
[22] P. Brucker and P. Brucker, "Scheduling algorithms", vol. 3, **(2007)**, Springer.
[23] B. Kruekaew and W. Kimpan, "Virtual Machine Scheduling Management on Cloud Computing Using Artificial Bee Colony", in Proceedings of the International MultiConference of Engineers and Computer Scientists, **(2014)**.
[24] S. Uma, K. R. Gandhi, E. Kirubakaran and E. Kirubakaran, "A hybrid PSO with dynamic inertia weight and GA approach for discovering classification rule in data mining", International Journal of Computer Applications, vol. 40, **(2012)**.
[25] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms", in Evolutionary Computation, 2001, Proceedings of the 2001 Congress on, **(2001)**, pp. 94-100.
[26] S. Xianjun, C. Zhifeng, Y. Jincai and C. CaiXia, "Particle Swarm Optimization with Dynamic Adaptive Inertia Weight", in Challenges in Environmental Science and Computer Engineering (CESCE), 2010 International Conference on, **(2010)**, pp. 287-290.
[27] X. -S. Yang and S. Deb, "Cuckoo search via Lévy flights", in Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, **(2009)**, pp. 210-214.
[28] X. -S. Yang and S. Deb, "Engineering optimisation by cuckoo search", International Journal of Mathematical Modelling and Numerical Optimisation, vol. 1, **(2010)**, pp. 330-343.
[29] T. Rambharose and A. Nikov, "Computational intelligence-based personalization of interactive web systems", **(2011)**.
[30] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. 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, **(2011)**, pp. 23-50.

