ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2016.0155

OPTIMIZED PARTICLE SWARM OPTIMIZATION BASED DEADLINE CONSTRAINED TASK SCHEDULING IN HYBRID CLOUD

Dhananjay Kumar[1], B. Kavitha[2], M. Padmavathy[3], B. Harshini[4], E. Preethi[5] and P. Varalakshmi[6]

[1,3,4,5]*Department of Information Technology, Anna University, MIT Campus, Chennai, India*

E-mail: [1]dhananjay@annauniv.edu, [3]padmit93@gmail.com, [4]mail2harshipriya@gmail.com, [5]preethi1381995@gmail.com

[2,6]*Department of Computer Technology, Anna University, MIT Campus, Chennai, India*

E-mail: [2]sridhar.kavitha@gmail.com, [6]varanip@gmail.com

Abstract

Cloud Computing is a dominant way of sharing of computing resources that can be configured and provisioned easily. Task scheduling in Hybrid cloud is a challenge as it suffers from producing the best QoS (Quality of Service) when there is a high demand. In this paper a new resource allocation algorithm, to find the best External Cloud provider when the intermediate provider's resources aren't enough to satisfy the customer's demand is proposed. The proposed algorithm called Optimized Particle Swarm Optimization (OPSO) combines the two metaheuristic algorithms namely Particle Swarm Optimization and Ant Colony Optimization (ACO). These metaheuristic algorithms are used for the purpose of optimization in the search space of the required solution, to find the best resource from the pool of resources and to obtain maximum profit even when the number of tasks submitted for execution is very high. This optimization is performed to allocate job requests to internal and external cloud providers to obtain maximum profit. It helps to improve the system performance by improving the CPU utilization, and handle multiple requests at the same time. The simulation result shows that an OPSO yields 0.1% - 5% profit to the intermediate cloud provider compared with standard PSO and ACO algorithms and it also increases the CPU utilization by 0.1%.

Keywords:

Hybrid Cloud, Particle Swarm Optimization, Ant Colony Optimization, Task Scheduling

1. INTRODUCTION

Cloud computing system is capable of lending infrastructure as a service to customers in such a way that both the provider and customer are equally benefitted [1]. The problem faced by the cloud providers is the need to cater to peak demands at any given instance. A solution to this problem is the use of hybrid cloud framework where private cloud providers can get the help of public cloud providers. The issue is how to allocate resources with maximum profit while guaranteeing QoS. The use of real infrastructures such as Amazon EC2, limits the scale of the infrastructure, and makes the reproduction of results an extremely difficult undertaking. The main reason for this condition prevailing in the Internet based environments is beyond the control of developers. This problem can be solved using an evolutionary concept of execution of Particle Swarm Algorithm. PSO can also be helpful when there are more than two datacenters exists or mapping of VMs are required [2]. It considers each variable as a particle and searches through the problem space to find an optimal solution.

The PSO algorithm has a strong ability to find the most optimistic result but it suffers from converging to local optimum. In self-adaptive learning PSO (SLPSO), four updating strategies are used to adaptively update the velocity of each particle considered thereby finding optimal solution. Experimental result [3] shows that, SLPSO can improve a cloud provider's profit by 0.25%-11.56% compared with standard PSO. However in SLPSO overhead occurs when the runtime of tasks is far from normal rates. Another metaheuristic algorithm, Ant colony optimization (ACO) can also be used for scheduling of resources. An ant is a simple computational agent in the ant colony optimization algorithm. It iteratively constructs a solution for the problem at hand. The intermediate solutions are referred to as solution states. At each iteration of the algorithm, each ant moves from a state *x* to state *y*, corresponding to a more complete intermediate solution. They have an advantage over simulated annealing and genetic algorithm approaches of similar problems when the graph may change dynamically. They can be run continuously and adapt to changes in real time. ACO also has the disadvantage of converging to local optimum in addition to convergence being complex. This paper proposes an optimized particle swarm optimization algorithm in combination with ant colony algorithm which can be used for the scheduling problem.

The first part makes use of the fast convergence of PSO to search the particles optimum position and make it as the start position of ants. The second part makes use of the merit of positive feedback and structure of solution set to search the global optimum scheduling. The results obtained have shown the proposed approach is feasible and effective for job scheduling problem. An issue faced in the use of hybrid cloud framework is the need for federation among cloud providers which in turn needs a framework or a set of policies to be established.To address those concerns, a hybrid cloud computing model which users may adopt as a cost-saving methodology to make use of public cloud services along with their privately-owned data centres is being used. As the core of this model, an intelligent workload factoring service is designed for proactive workload management. It enables federation between on-and-off premise infrastructures for hosting Internet-based applications. It involves segregation of baseworkload and flash crowdworkload, the two naturally different components composing the application workload. The core technology of the intelligent workload factoring service is a fast frequent data item detection algorithm, which enables factoring incoming requests, upon changing application data popularity [4].Here the disadvantages include lack of security management, consistency management and data replication. This paper avoids the necessity of cloud federation and allows the private cloud providers to directly outsource their requests to public cloud providers without any inter cloud agreement. It also aims to improve CPU utilization and reduce runtime for high range of tasks.

2. PSO ALGORITHM

Particle swarm optimization (PSO) [13] is an optimization algorithm which simulates the movement and flocking of birds. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions called particles fly through the problem space by following the current optimum particles. PSO optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. PSO optimizes a problem by having a population of candidate solutions, particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position but, is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions. The choice of PSO parameters can have a large impact on optimization performance. Selecting PSO parameters that yield good performance has therefore been the subject of much research.

In standard PSO, each individual in the swarm is treated as a particle in a D-dimensional search space, and represented by a three tuple $\{X_i, V_i, P_i\}$, where $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ and $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ denote the position and velocity of particle i, respectively, and $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ represents the personal best (*pbest*) of particle i (that is, the best position achieved by particle i). $G = (g_1, g_2, \ldots, g_d)$ denotes the global best (*gbest*), namely the best position tracked by the entire swarm. The value of each element in the vector V_i can be clamped to the range of $[-v_{max}, v_{max}]$ to control the excessive roaming of particle outside the search, and updated by

$$V_{id}(t+1) = \omega V_{id}(t) + c_1 r_1 [X_{id}(t) - p_{id}(t)] + c_2 r_2 [X_{id}(t) - g_d(t)] \quad (1)$$

where, $i = 1, 2, \ldots, M$ denotes the number of particles and $d = 1, 2, \ldots, D$ is the dimension of particles. r_1 and r_2 are the uniformly distributed random number whose range is [0, 1]. c_1 and c_2 are learning factors. c_1 is the individual cognition component, representing the search ability of the particle itself, and is the social communication component representing the influence from the social environment. ω is the inertia weight to avoid unlimited growth of particle's velocity. The particle flies toward a new position, and each value of the new position should not exceed the range of [min X, max X].

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1) \quad (2)$$

In Eq.(1) and Eq.(2), the ω (moment of inertia) which takes value of 0.4 and the two cognition learning factors c_1, c_2 takes the value of 2.0. At the beginning, the position and velocity of each particle in the swarm are initialized randomly. Then, each particle is led by its own flying experience (*pbest*) and the best particle (*gbest*), i.e., updated by Eq.(1) and Eq.(2). This process is repeated until a user-defined stopping criterion is reached. The steps of standard PSO is as follows:

Step 1: Randomly initialize position and velocity of all particles.

Step 2: Evaluate the profit of all particles; let each particle's *pbest* be its current position; let *gbest* be the best one among all particles.

Step 3: Updated each particle's velocity and position using Eq.(1) and Eq.(2).

Step 4: Calculate the profit values of all particles.

Step 5: Update *pbest*. For each particle, if the profit value of its new position is better than that of its *pbest*, then replace its *pbest* by the new position.

Step 6: Update *gbest*. For each particle, if the profit value of its new position is better than that of the *gbest*, then replace the *gbest* by the new position.

Step 7: If all the iterations gets completed or the profit obtained for all task is maximum or if solution for all tasks converges for at least two generations, then output *gbest* and its profit value; otherwise, go to Step 3.

3. ANT COLONY OPTIMIZATION

Ant colony optimization (ACO) [14] takes inspiration from the foraging behaviour of some ant species. These ants deposit pheromone on the ground in order to mark some favourable path that should be followed by other members of the colony. Ant colony optimization exploits a similar mechanism for solving optimization problems. In ACO, a number of artificial ants build solutions to the considered optimization problem at hand and exchange information on the quality of these solutions via a communication scheme that is reminiscent of the one adopted by real ants. Ants are social insects. They live in colonies and their behaviour is governed by the goal of colony survival rather than being focused on the survival of individuals. The behaviour that provided the inspiration for ACO is the ant's foraging behaviour, and in particular, how ants can find shortest paths between food sources and their nest. When searching for food, ants initially explore the area surrounding their nest in a random manner. While moving, ants leave a chemical pheromone trail on the ground.

Ants can smell pheromone. When choosing their way, they tend to choose, in probability, paths marked by strong pheromone concentrations. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the quantity of pheromone that an ant leaves on the ground may depend on the quantity and quality of the food. The pheromone trails will guide other ants to the food source. The indirect communication between the ants via pheromone trails known as stigmergy enables them to find shortest paths between their nest and food sources. The main differences between the behaviour of the real ants and the behaviour of the artificial ants are as follows:

1. While real ants move in their environment in an asynchronous way, the artificial ants are synchronized, i.e. at each iteration of the simulated system; each of the artificial ants moves from the nest to the food source and follows the same path back.
2. While real ants leave pheromone on the ground whenever they move, artificial ants only deposit artificial pheromone on their way back to the nest.
3. The foraging behaviour of real ants is based on an implicit evaluation of a solution (i.e., a path from the nest to the food source). By implicit solution evaluation we mean the fact that shorter paths will be completed earlier than longer ones, and therefore they will receive pheromone

reinforcement more quickly. In contrast, the artificial ants evaluate a solution with respect to some quality measure which is used to determine the strength of the pheromone reinforcement that the ants perform during their return trip to the nest.

In general, the ACO approach attempts to solve an optimization problem by iterating the following:

1. Candidate solutions are constructed using a pheromone model, that is, a parameterized probability distribution over the solution space.
2. The candidate solutions are used to modify the pheromone values in a way that is deemed to bias future sampling towards high-quality solutions. The pheromone update aims to concentrate the search in regions of the search space containing high-quality solutions. It implicitly assumes that good solutions consist of good solution components.

3.1 ALGORITHM

Step 1: Set parameters; initialize pheromone trails.

Step 2: Check whether number of iterations has exceeded or has obtained the maximum profit.

Step 3: If Step 2 conditions are not met, then

i. Construct the Ant Solutions by using Eq.(3)

ii. Apply the Local Search by using Eq.(1) and Eq.(2)

iii. Update Pheromones.

Step 4: Else terminate the algorithm by displaying the allocated task to thefeasible resource.

In the construction of a solution, ants select the following node to be visited through a stochastic mechanism. When ant k is in node i and has so far constructed a partial solution, the probability of going to node j is given by:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{c_{i1}\in N(s^{p})}\tau_{ij}^{\alpha}\eta_{ij}^{\beta}} & \text{if } c_{ij} \in N(s^{p}), \\ \text{otherwise} \end{cases} \quad (3)$$

where,

$N(s^p)$ - set of feasible components

α and β control the relative importance of pheromone

τ - pheromone concentration

The heuristic information η_{ij}

$\eta_{ij} = 1/d_{ij}$

d_{ij} - profit obtained

4. OPSO ALGORITHM

The OPSO algorithm is a combination of ACO algorithm with PSO algorithm, which can solve many optimization problems. But both have the disadvantage of converging at local optimum. This OPSO algorithm combines both algorithms to find optimum solution in global search space. The proposed algorithm updates the global pheromone, *pbest* and *gbest* at each iteration. Stopping criteria are if maximum number of iterations is achieved or when the optimum solution has appeared.

4.1 PROBLEM PARAMETERS

The OPSO algorithm uses the Eq.(4), Eq.(5), Eq.(6), Eq.(7) to initialize and update the pheromone.

1. CPU pheromone

$$\tau_{ic}(0) = ((n * c)/(n_0 * c_0)) * 100\% \quad (4)$$

2. Memory pheromone

$$\tau_{im}(0) = (m/m_0) * 100\% \quad (5)$$

3. External memory pheromone

$$\tau_{ie}(0) = e/e_0 * 100\% \quad (6)$$

$$\tau_{ij}(0) = a\,\tau_{ic}(0) + b\,\tau_{ie}(0) + c\,\tau_{im}(0)\ (a + b + c = 1) \quad (7)$$

where,

n = number of CPU

c(MIPS) = processing power

m = memory capacity

$m_{max} = m_0$, $n_{max} = n_0$, $c_{max} = c_0$, $e_{max} = e_0$

4.2 ALGORITHM

Step 1: Initialize the hardware pheromone of node using Eq.(4), Eq.(5), Eq.(6). The total pheromone in each node is calculated by Eq.(7).

Step 2: Put '*a*' number of ants on '*n*' nodes randomly and calculate the profit value for each ant.

Step 3: Calculate the *pbest* and *gbest* for epoch 0. The velocity of each particle is updated by using Eq.(1) and the solution is updated by using strategy given in Eq.(2).

Step 4: Move to the next resource *j* according to the probability P_{ij} which is given by Eq.(3).

Step 5: Update the velocity, position and pheromone at each epoch and find *pbest* and *gbest* for each ant and choose the next resource based on Step 4.

Step 6: If *pbest* for all the ants converges or if no. of iterations reaches the maximum the algorithm stops.

Step 7: Output the optimal solution or resource allocated for each task.

The algorithm initializes the fitness function (maximizing the profit) and concentration of pheromone initially in each resource. Randomly assign the profit for each task as position and velocity. Evaluate profit for next resource and based upon the probability move to next resource. Finally update the pheromone for each resource, update the new profit obtained for each task and update velocity and position. Check whether the iteration count has been exceeded its limit or the solution found converges for at least two iterations or the fitness function is satisfied. If yes then terminate the algorithm, else continue the same process until the best feasible solution with high profit is obtained.

5. IMPLEMENTATION AND RESULTS

The required cloud environment is set up in cloudsim software [17]. The Hybrid cloud is set up using a private cloud with a datacentre consisting of three virtual machines and three public cloud providers with a datacentre each. Each of the public cloud provider has different system configuration and different VM

types. Each public cloud provider has three types of VM one is small, one is medium and the other one is large. The Table.1 list the prices in rupees of external cloud provider [3] which is used to create cloudsim simulation environment. Each VM in the public cloud are assigned cost and price.

The Table.2 specifies the cost and price in rupees for the internal/private cloud [3] used in implementation of Hybrid cloud. The VM's in private cloud ranges between small, medium and large.

The configuration specifications for each type of VM used in implementation of the Hybrid cloud is listed in Table.3 [3]. The small, medium, and large VM types has different configuration of CPU (MIPS) and memory.

Table.1. Price of external cloud provider

Size of VM	EC A	EC B	EC C
Small	0.085	0.070	0.100
Medium	0.34	0.30	0.40
Large	0.68	0.70	0.72

Table.2. Cost and price of internal cloud provider

	Small	Medium	Large
Cost	0.03	0.12	0.24
Price	0.08	0.32	0.64

Table.3. VM configuration specifications

	CPU(MIPS)	Memory(GB)
Small	20	40
Medium	512	1024
Large	1024	2048

After the required environment has been set up the proposed algorithm optimized PSO (OPSO) algorithm is run in this environment and the result obtained is noted. The proposed algorithm is run with different number of tasks, 50, 100, 1000 tasks and with varying amount of runtime of tasks. The same number of tasks is run using Particle Swarm Optimization algorithm, Ant colony Optimization, Self-adaptive Learning Particle Swarm Optimization and the performance is recorded.

5.1 PERFORMANCE ANALYSIS

The proposed OPSO algorithm is deployed in hybrid cloud in cloudsim environment [17] and performance measure of respective approaches were observed. The parameter measured for performance analysis are the profit, CPU utilization, average runtime. The simulation were run for several iterations on an Intel Pentium dual core in a windows 8 environment. The OPSO algorithm yield 0.1% - 5% profit to the cloud provider compared with standard PSO and ACO algorithms and it also increases the CPU utilization by 0.1%.

The net profit were computed while executing ACO, PSO, SLPSO and ACOPSO algorithms for 50 numbers of tasks (Fig.1). ACOPSO algorithm outperforms the other algorithm in obtaining high profit for intermediate cloud provider. For smaller number of tasks it's easy to find the best resource as the OPSO algorithm converges at a faster rate. The solution search space scope is widened by ACO algorithm and the best solution in the search space is found out by the PSO algorithm.

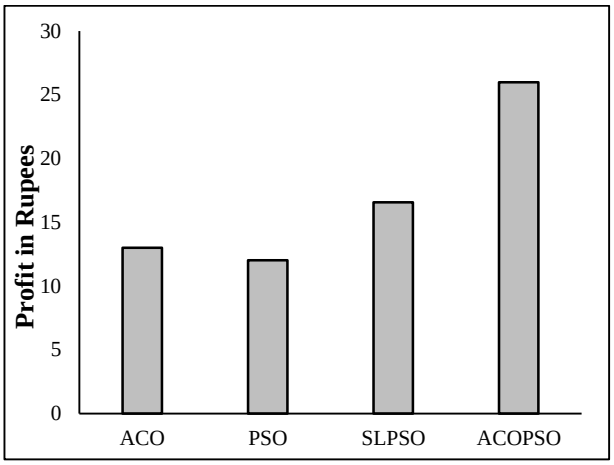

Fig.1. Profit for 50 tasks

The Profit computation was again repeated for 100 and 1000 number of tasks (Fig.2). ACOPSO algorithm gives a better profit when compared with other algorithm for larger number of tasks. The profit increases as the utilization of CPU increases; more number of tasks gets completed within a given time. Search space gets widened by ACO algorithm and more optimal solution is found at a faster pace.

The average CPU utilization were calculated while executing ACO, PSO, SLPSO and ACOPSO algorithms for 50, 100 and 1000 number of tasks (Fig.3). The average CPU utilization computation reveals that the ACOPSO algorithm increases the CPU utilization rate by 0.1% when compared with other algorithms. Due to the high convergence rate the number of tasks gets completed in a unit time increases. The CPU ideal time decreases which results in decrease of tasks runtime.

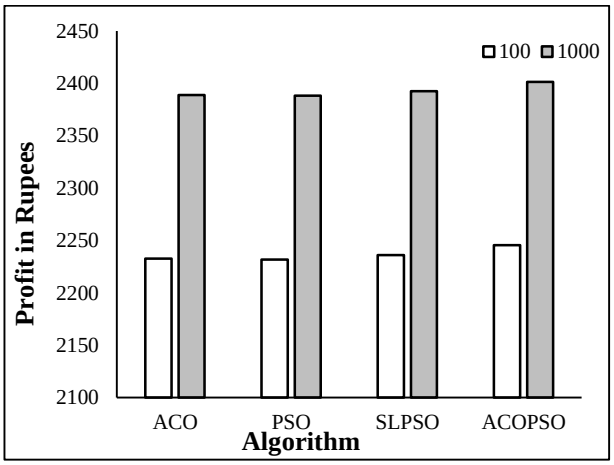

Fig.2. Profit for 100 and 1000 tasks

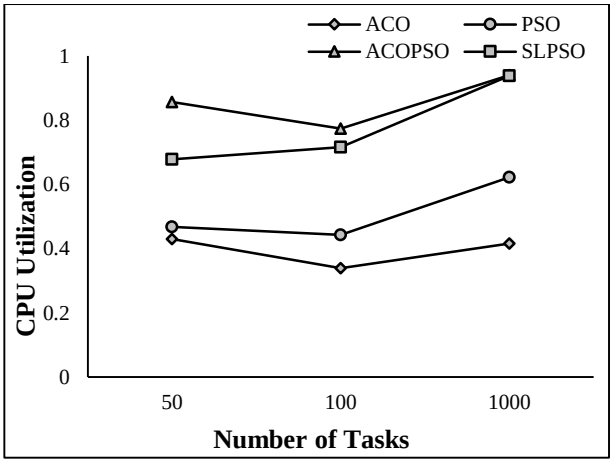

Fig.3. CPU utilization of ACO, PSO, ACOPSO and SLPSO

The average runtime were computed while executing ACO, PSO, SLPSO and ACOPSO algorithms for 50, 100 and 1000 number of tasks (Fig.4). The average runtime obtained for ACOPSO algorithm decreases to a greater extent when compared to other algorithms. The run time decreases, because the more effective solution is found out by the Optimised PSO algorithm from the solution search space. The waiting time and turnaround time of each task is reduced to a greater extent.

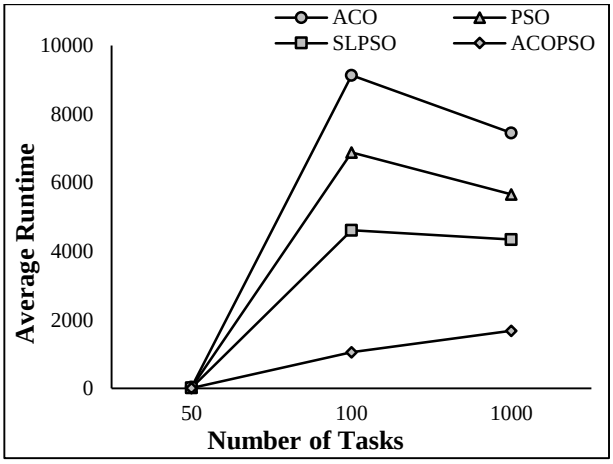

Fig.4. Runtime of simulation

6. CONCLUSION

Cloud services are being widely used in different domains. Hence the cloud service providers are increasingly facing high demand of requests from customers. They are forced to maintain quality of service while meeting peak demands.

A solution to this problem is the use of hybrid cloud framework where private cloud providers can outsource their requests to public cloud providers. The proposed OPSO algorithm allocates the requests to private cloud providers and public cloud providers such that maximum profit is obtained in both the cases. An optimized PSO which is a combination of ACO and PSO approach has been formulated in order to effectively allocate resources. Optimized PSO yielded 0.1% - 5% profit to the cloud provider compared with standard PSO and ACO algorithms and it also increases the CPU utilization by 0.1%. The average runtime for the proposed algorithm has a drastic decrease compared to other algorithms. The proposed solution is designed to guarantee user level QoS and improve IaaS provider's credibility and economic benefit.

The proposed work can be further extended to improve the efficiency of the OPSO approach by combining with a Hypergraph [4] clustering or *k* means type clustering [18] of requests and resources before applying the OPSO approach in order to further increase the speed of OPSO.

ACKNOWLEDGEMENT

We would like to express our sincere thanks and deep sense of gratitude to the Department of Information Technology, Anna University, MIT Campus, Chennai for providing necessary infrastructure and support to complete our research work.

REFERENCES

[1] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg and Ivona Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype and Reality for Delivering Computing as the 5th Utility", *Future Generation Computer Systems*, Vol. 25, No. 6, pp. 599-616, 2009.

[2] S. Bhardwaj, L. Jain and S. Jain, "Cloud Computing: A Study of Infrastructure as a Service (IaaS)", *International Journal of Engineering and Information Technology*, Vol. 2, No. 1, pp. 60-63, 2010.

[3] Xingquan Zuo, Guoxiang Zhang and Wei Tan, "Self-Adaptive Learning PSO-Based Deadline Constrained Task Scheduling for Hybrid IaaS Cloud", *IEEE Transactions on Automation Science and Engineering*, Vol. 11, No. 2, pp. 564-573, 2014.

[4] Hui Zhang, Guofei Jiang, Kenji Yoshihira and Chen Haifeng, "Proactive Workload Management in Hybrid Cloud Computing", *IEEE Transactions on Network and Service Management*, Vol. 11, No. 1, pp. 90-100, 2014.

[5] Li Li, Wang Keqi and Zhou Chunnan, "An Improved Ant Colony Algorithm Combined with Particle Swarm Optimization Algorithm for Multi-objective Flexible Job Shop Scheduling Problem", *Proceedings of International Conference on Machine Vision and Human-Machine Interface*, pp. 88-91, 2010.

[6] Yu Wang, Bin Li, Thomas Weise, Jianyu Wang, Bo Yuan and Qiongjie Tian, "Self-Adaptive Learning Based Particle Swarm Optimization", *Information Sciences*, Vol. 181, No. 20, pp. 4515-4538, 2011.

[7] Lu Huang, Hai-shan Chen and Ting-ting Hu, "Survey on Resource Allocation Policy and Job Scheduling Algorithms of Cloud Computing" *Journal of Software*, Vol. 8 , No. 2, pp. 480-487, 2013.

[8] M.A. Tawfeek, A. El-Sisi, A.E. Keshk and F.A. Torkey, "Cloud Task Scheduling Based on Ant Colony Optimization", *Proceedings of 8th International Conference on Computer Engineering and Systems*, pp. 64-69, 2013.

[9] D. Bruneo, "A Stochastic Model to Investigate Data Centre Performance and Qos in IaaS Cloud Computing Systems",

IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 3, pp. 560-569, 2014.

[10] N.D. Doulamis, P. Kokkinos and E. Varvarigos, "Resources Selection for Task with Time Requirements Using Spectral Clustering", *IEEE Transactions on Computers*, Vol. 63, No. 2, pp. 461-474, 2014.

[11] Haiying Shen and Guoxin Liu, "An Efficient and Trustworthy Resource Sharing Platform for Collaborative Cloud Computing", *IEEE Transactions on Distributed and Parallel Systems*, Vol. 25, No. 4, pp. 862-875, 2014.

[12] Junwei Cao, Keqin Li and I. Stojmenovic, "Optimal Power Allocation and Load Distribution for Multiple Heterogeneous Multicore Server Processors across Clouds and Data Centers", *IEEE Transactions on Computers*, Vol. 63, No. 1, pp. 45-58, 2014.

[13] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", *Proceedings of IEEE International Conference on Neural Networks*, Vol. 4, pp. 1942-1948, 1995.

[14] M. Dorigo, M. Birattari and T. Stutzle, "Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique", *IEEE Computational Intelligence Magazine*, Vol. 1, No. 4, pp. 28-39, 2006.

[15] A. Ratnaweera, S. Halgamuge and H.C. Watson, "Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients", *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 3, pp. 240-255, 2004.

[16] Hu Xiaohui and R. Eberhart, "Multiobjective Optimization using Dynamic Neighborhood Particle Swarm Optimization", *Proceedings of Congress on Evolutionary Computation*, Vol. 2, pp. 1677-1681, 2002.

[17] Kavita Bhatt and Mahesh Bundele, "CloudSim Estimation of a Simple Particle Swarm Algorithm", *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 3, No. 8, pp. 1279-1287, 2013.

[18] A. Ahmadyfard and H. Modares, "Combining PSO and K-means to Enhance Data Clustering", *Proceedings of International Symposium on Telecommunications*, pp. 688-691, 2008.