International Journal on Applications in Information and Communication Engineering
Volume 3: Issue 3: September 2017, pp 1-9 . www.aetsjournal.com ISSN (Online) : 2394-6237

IPSO ALGORITHM FOR WORKFLOW SCHEDULING IN CLOUD COMPUTING ENVIRONMENTS

Dr. N. SADHASIVAM

Abstrac t— Cloud computing is a type of Internet-based computing that provides computing resources to store manage and process data over internet. It is a computing platform that resides in a service provider's large data centre. It is a dynamic environment which provides the services typically Infrastructure as a Service, Software as a Service and Platform as a Service .These sservices are able to address a wide range of needs of clients. Workflow scheduling is a major factor that influences the performance of system in a cloud computing environment. The cloud service providers and consumers have different objectives and requirements. For the moment, the load and availability of the resources vary dynamically with time. Workflow scheduling discovers resources and allocates tasks on suitable resources. Workflow scheduling plays an important role in the workflow management. Scheduling problems belong to a broad class of optimization problem. It aimed at finding an optimal matching of tasks to different sets of resources. The primary objective of this work is to derive the Improved Particle swarm optimization approach for mapping the tasks to the computer resources such that the total cost is minimized.

Keywords — Workflow Scheduling, Cloud Computing Environment, Scheduling Algorithm, Optimization, PSO, IPSO.

I. INTRODUCTION

Computing at the scale of the cloud allows users to access supercomputer-level power. Instead of operating their own data centres, firms might rent computing power and storage capacity from a service provider, making them pay only for what they use, as they do with electricity or water. The paradigm of cloud computing has also been referred to as "utility computing," in which computing capacity is treated like any other metered utility service-one pays only for what one uses. Users can reach into the cloud for resources as they need from anywhere at anytime. For this reason, cloud computing has also been described as "on-demand computing". It is provided as a service by another company and accessed over the Internet, usually in a completely seamless way. Exactly where the hardware and software are located and how they all work do not matter to users. For the user it is just somewhere up in the nebulous "cloud" that the Internet represents.

Dr. N. SADHASIVAM , Assistant Professor (Senior Grade), Department of Computer Science and Engineering, Bannari Amman Institute of Technology, Erode, India . (E-mail: sadhasivamn82@gmail.com)

Cloud computing was coined for what happens when applications and services are moved into the internet "cloud." Cloud computing is not something that suddenly appeared overnight; in some form it may be traced back to a time when computer systems remotely time-shared computing resources and applications. More currently though, cloud computing refers to the many different types of services and applications being delivered in the internet cloud. In many cases, the devices used to access these services and applications do not require any special applications.

Task scheduling is a major concern which greatly influences the performance of cloud computing environment. The cloud service providers and consumers have different objectives and requirements. For the moment, the load and availability of the resources vary dynamically with time. Therefore, in the cloud environment scheduling resources is a complicated problem. Moreover, task scheduling algorithm is a method by which tasks are allocated or matched to data center resources. However, absolutely perfect scheduling algorithms do not exist because of conflicting scheduling objectives (Pandey et al. 2010).

Workflow scheduling is the problem of mapping each task to appropriate resource and allowing the tasks to satisfy some performance criterion. A workflow consists of a sequence of concatenated (connected) steps. The workflow enables the structuring of applications in a directed acyclic graph form where each node represents the task and edges represent the dependencies between the nodes of the applications.

A single workflow consists of a set of tasks and each task communicates with another task in the workflow. Workflows are supported by Workflow Management

Systems (WMS). Workflow scheduling discovers resources and allocates tasks on suitable resources. Workflow scheduling plays a vital role in the workflow management. Proper scheduling of workflow can have an efficient impact on the performance of the system. However, for proper scheduling of workflows, various scheduling algorithms are used.

Optimization is essentially everywhere from engineering design to economics and from holiday planning to Internet routing. As money resources and time are always limited, the optimal utility of these available resources is crucially important. Many engineering optimization problems are usually quite difficult to solve and many applications, have to deal with these complex problems. In these problems, search space grows exponentially with the problem size. Therefore the traditional optimization methods do not provides a suitable solution for them. Hence, over the past few decades, many metaheuristic algorithms had been designed to solve such problems.

Researchers have shown good performance of metaheuristic algorithms in a wide range of complex problems such as scheduling, data clustering, image and video processing, tuning of neural networks, pattern recognition etc. Algorithms with stochastic components were often referred to as heuristic in the past, though the recent literature tends to refer to them as metaheuristics. It seems to be advisable to follow Glover's convention and call all modern nature inspired algorithms metaheuristics (Glover 1986, Glover & Kochenberger 2003). Loosely speaking, heuristic means to find or to discover by trial and error. Here meta means beyond or higher level, and metaheuristics generally perform better than simple heuristics.

The word "metaheuristic" was coined by Fred Glover in his seminal paper (Glover 1986) and a metaheuristic can be considered as a "master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality". In addition all metaheuristic algorithms use a certain trade off of randomization and local search. Quality solutions to difficult optimization problems can be found in a reasonable amount of time, but there is no guarantee that optimal solutions can be reached. It is hoped that these algorithms work most of the time, but not all the time. Almost all metaheuristic algorithms tend to be suitable for global optimization. In this connection, it seems relevant to recall the excellent review that Voss (2001) given.

Two major components of any metaheuristic algorithms are: intensification and diversification or exploitation and exploration (Blum & Roli 2003). Diversification means to generate diverse solutions so as to explore the search space on a global scale, while intensification means to focus the search in a local region knowing that a current good solution is found in this region. A good balance between intensification and diversification should be found during the selection of the best solutions to improve the rate of algorithm convergence. The selection of the best ensures that solutions will converge to the optimum, while diversification via randomization allows the search to escape from local optima and at the same time, increases the diversity of solutions. A good combination of these two major components will usually ensure that global optimality is achievable.

A workflow application is a graph G=(V,E) that can be represented by a Directed Acyclic Graph (DAG), where V is the set of n tasks {T1, T2,......,Tn} and E is a set of e edges, that represents the dependencies. Each Ti ∈ V, represents a task in the application and each edge (ei..........ej) ∈ E represents a precedence constraint, such that the execution of Tj ∈ V cannot be started before Ti ∈V finishes its execution. A task with no parent is known as an entry or root task and a task with no children is known as exit or last task.

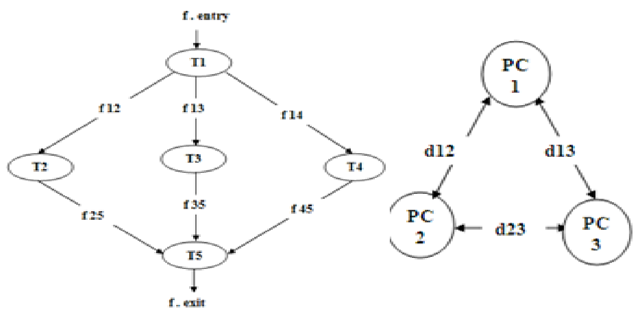

Figure 1.1 Sample workflow & compute resources (PC)

Consider a set of compute resources PC = {1, ..., j} and a set of tasks T = {1, ..., k}. The cost of computation of a task on a compute host is inversely proportional to the time it takes for computation on that resource. Then, the time it takes for computation on that resource is inversely proportional to the cost of computation of a task on a compute host. Consider that the cost of unit data access $d_{i,j}$ from a resource i to a resource j is known. The transfer cost can be calculated according to the bandwidth between the resources.

Fig.1.1 depicts the workflow structure with five tasks (Pandey et al. 2010), which are represented as nodes. The dependencies among tasks are represented by arrows. The entry task may have an input file (e. g. f.entry) and the exit task produces the output file (e. g.

f.exit). Each task generates output data after it has completed (f_{12}, f_{13}, f_{14}..., f_{ij}). The task's children use these data, if any. Three computer sources (PC1, PC2 and PC3) interconnected with varying bandwidth and having its own servers (S1, S2, S3). Therefore, the main objective is to assign the workflow tasks to the computer sources so that the total cost of computation is minimized.

II. PROBLEM FORMULATION

The problem can be formulated to identify a task-resource mapping instance P, such that when calculating the total cost incurred using each compute resource PC, the high cost among all the compute resources is minimized. Let $C_{exe}(P)_j$ be the total cost of all the tasks assigned to a compute resource PC_j. The total cost value is computed by summarizing all the node weights of all tasks assigned to each resource in the mapping M. Let $C_{tx}(P)_j$ be the total access cost between tasks assigned to a compute resource PC_j. The $C_{tx}(P)_j$ is the product of the output file size from task k_1 to task k_2 and the cost of communication from the resource where k_1 is mapped ($P(k_1)$) to another resource where k_2 is mapped ($P(k_2)$).

The average cost of communication of unit data between two resources is given by $dP(k_1)$, $dP(k_2)$.

$$C_{exe}(P)_j = \sum_{k} w_{kj} \qquad \forall P(K) = j \tag{1.1}$$

$$C_{tx}(P)_j = \sum_{k1 \in T} \sum_{k2 \in T} d_{P(k1),P(k2)} e_{k1k2} \, \forall P(K) = j$$

$$\forall P(k_1) = j \; and \; P(k_2) \neq j \tag{1.2}$$

$$C_{total}(P)_j = C_{exe}(P)_j + C_{tx}(P)_j \tag{1.3}$$

$$Cost(P) = max(C_{total}(P)_j) \, \forall j \in P \tag{1.4}$$

$$Minimize(Cost(P) \quad \forall P) \tag{1.5}$$

Equation 1.4 implies that all the tasks are not mapped to single compute resource. Initial cost maximization will distribute tasks to all the resources. Subsequent minimization of the overall cost indicates that the total cost is minimal even after initial distribution. For a given instance P, the total cost $C_{total}(P)_j$ for a compute resource PC_j is the sum of execution cost and access cost. When estimating the total cost for all the resources, the largest cost for all the resources is minimized. It indirectly ensures that the tasks are not mapped to a single resource and there will be a distribution of cost among the resources.

III. OBJECTIVE

Metaheuristic techniques are high-level frameworks which utilize heuristics in order to find solutions to combinatorial optimization problems at a finite computational cost. This kind of problem may be classified as NP-hard or NP-complete or be a problem for which a polynomial time algorithm is known to exist but is not practical. In general, workflow applications are represented as a directed acyclic graph. In general, the mapping of jobs to the computer sources is an NP-complete problem. The primary objective of this work is to derive the metaheuristic approaches for mapping the tasks to the compute resources such that the total cost of computation is minimized.

IV. TASK - RESOURCE ENCODING

The metaheuristic algorithm starts with random initialization of solutions. In this problem, the solutions are the tasks to be assigned and the dimensions of the solutions are the number of tasks in a workflow. The value assigned to each dimension of a solution is the computing resources indices. Thus the solution represents a mapping of resource to a task. Fig.1.2 shows the solution representation for the workflow.

T1	T2	T3	T4	T5
PC1	PC2	PC2	PC3	PC1

Figure 1.2 Solutions representation for the workflow

V. LITERATURE REVIEW

Tsai et al. (2014) implemented Hyper-Heuristic Scheduling Algorithm (HHSA) for providing effective cloud scheduling solutions. The diversity detection and improvement detection operators are utilized in this approach dynamically to determine better low-level heuristic for the effective scheduling. HHSA can reduce the makespan of task scheduling and improves the overall scheduling performance. The drawback is that the approach has high overhead of connection which reduces the importance of scheduling and thus reduces the overall performance.

Zhu et al. (2014) proffered real-time task oriented Energy Aware (EA) scheduling called EARH for the virtualized clouds. The proposed approach is based on Rolling-Horizon (RH) optimization and the procedures are developed for creation, migration, and cancellation of VMs dynamically to adjust the scale of cloud to achieve real time deadlines and reduce energy. The EARH approach has the drawback of the number of cycles assigned to the VMs that cannot be updated dynamically.

Zuo et al. (2014) produced a Self-adaptive Learning Particle Swarm Optimization (SLPSO) based scheduling approach for deadline constraint task scheduling in hybrid Infrastructure as a Service (IaaS) clouds. The approach solves the problem of meeting the peak demand for preserving the quality-of-service constraints by using the PSO optimization technique. The approach provides better scheduling of the tasks by maximizing the profit of IaaS provider while guaranteeing QoS. The problem with this approach is the lack of priority determination which results in failure of deadline tasks. Thus scheduling tasks in a cloud computing environment is a challenging process.

Zhu et al. (2016) advanced an Evolutionary Multi-Objective (EMO) workflow scheduling approach to reduce the workflow scheduling problem such as cost and makespan. Due to the specific properties of the workflow scheduling problem, the existing genetic operations, such as binary encoding, real valued encoding and the corresponding variation operators are based on them in the EMO. The problem is that the approach does not consider monetary costs and time overheads of both communication and storage.

Pandey et al. (2010) expounded a PSO-based heuristic algorithm for dynamic scheduling of the data intensive workflow applications, where the size and quantity of the data are large. To transfer and store the data as compared to the execution of tasks, more time is needed. This scheme optimizes the cost of the task-resource mapping based on the solution given by the PSO and takes both computation cost and data transmission cost into account.

Guo et al. (2012) formulated a model for task scheduling and mooted a Particle Swarm Optimization (PSO) algorithm which is based on small position value rule to minimize the cost of the processing. By virtue of comparing PSO algorithm with the PSO algorithm embedded in crossover and mutation and in the local research, the experiment results show that the PSO algorithm not only converges faster but also runs faster than the other two algorithms in a large scale. The experiment results prove that the PSO algorithm is more suitable to cloud computing.

Yang et al. (2013) recommended a PSO-based algorithm to solve task scheduling and resource allocation in cloud computing. The problem is to assign each subtask to an appropriate resource and to sequence the subtasks on the resources in order to achieve the objectives of this scheme. To formulate the problem, cloud user tasks can denote the set of *n* independent jobs, and each subtask is allowed to be processed on any given available resources. A subtask is processed on one resource at a time and the given resources are available continuously. This scheme shows that the PSO based fitness function is more effective and efficient with shorter completion time and lower cost.

Huang et al. (2013) set forth a scheme for workflow scheduling to minimize both the total cost and makespan. They present a PSO-based heuristics to realize the optimal mapping for the tunable objective. In this scheme, all the ready tasks assigned to a specific resource are independent, and it will speed up the workflow by scheduling the "bottleneck" task first, i.e. the task having most descendants. Thus, the ready tasks are sorted according to the number of descendants. If there is a tie, the one with a short execution time will be given a high priority to execute first.

Sadhasivam,N & Thangaraj (2017) proposed an IPSO algorithm to minimize the total cost for compute resource while mapping tasks to suitable resources.

VI. SYSTEM ANALYSIS

1) EXISTING SYSTEM

Particle swarm optimization (PSO) is a population based optimization technique - inspired by social behavior of bird flocking. PSO This swarm behaviour leads to the assumption that information is owned jointly in the flocking. Initially, the swarm has a population which is random solutions. Each potential solution is represented as a particle (agent) and is given a random velocity and is flown through the problem space. Each and every particle has memory and each particle keeps track of its previous best position (p_{best}) and the corresponding fitness value. The swarm has another value called (g_{best}), which is the best value of all particles' p_{best}. It has been shown to be extremely effective in solving a wide range of engineering problems and solves them very quickly.

The PSO algorithm updates the velocity and position of each particle by the following equations (3.1) and (3.2) respectively.

$$V_{id}^{'} = \omega V_{id} + c_1 rand1()(P_{idb} - X_{id}) + c_2 rand2()(P_{gdb} - X_i \quad (3.1)$$

$$X_{id}^{'} = X_{id} + V_{id}^{'} \quad (3.2)$$

Where, c_1 and c_2 are the learning factors which determine the relative influence of cognitive and social component respectively. The *rand1()* and *rand2()* are uniformly distributed random numbers in the range from 0 to 1. V_{id}, X_{id} and P_{idb} are the velocity, position and the personal best of i^{th} particle in D^{th} dimension. The P_{gdb} is the global best of the swarm in D^{th} dimension.

2) PROPOSED SYSTEM

In the standard PSO algorithm, the convergence speed of particles is fast, but the adjustments of cognition component and social component make particles search around entire solution. According to velocity and position renewal formula, once the best individual in the swarm is trapped into a local optimum, the information sharing mechanism in PSO will attract other particles to approach this local optimum gradually and in the end, the whole swarm will be converged at this position. But according to velocity and position renewal equation (3.3) and (3.4), once the whole swarm is trapped into a local optimum, its cognition component and social component will become zero in the end; still, because *0<ω<1* and with the increased number of iterations, the velocity of particles will become zero in the end. Thus the whole swarm is hard to jump out of the local optimum if there is no way to achieve the global optimum.

$$V_{id}^{'} = \omega V_{id} + \eta_1 rand()(P_{idb} - X_{id}) + \eta_2 rand()(P_{gdb} - X_{id})$$

$$X_{id}^{'} = X_{id} + V_{id}^{'} \qquad (3.4)$$

Here a fatal weakness may result from this characteristic. With constant increase of iterations, the velocity of particles will gradually diminish and reach zero in the end. At this time, the whole swarm will be converged at one point in the solution space, if P_{gdb} particles haven't found P_{gdb}, the whole swarm will be trapped into a local optimum and the capacity of swarm jump out of a local optimum is rather weak. In order to get through this disadvantage, this work presented a new algorithm based on PSO. In order to avoid being trapped into a local optimum, the new PSO adopts a new information sharing mechanism. It is known that when a particle is searching in the solution space, it does not know the exact position of the optimum solution. But one can not only record the best positions an individual particle and the whole swarm have experienced, one can also record the worst positions an individual particle and the whole swarm have experienced. Thus we may make individual particles move in the direction of evading the worst positions an individual particle and the whole flocks have experienced, this will surely enlarge the global searching space of particles and enable them to avoid being trapped into a local optimum too early. At the same time, it will improve the possibility of finding global best in the searching space. In the new strategy, the particle velocity and position renewal formula are as follows:

$$V_{id}^{'} = \omega V_{id} + \eta_1 rand()(X_{id} - P_{idw}) + \eta_2 rand()(X_{id} - P_{gdw}) \qquad (3.5)$$

$$X_{id}^{'} = X_{id} + V_{id}^{'} \qquad (3.6)$$

Here, P_{idw}, P_{gdw} represent the worst position particle *id* has found and the worst positions of the whole swarm has found.

In standard PSO algorithm, the next flying direction of each particle is nearly determined; it can fly to the best individual and the best individuals for the whole swarm. In order to decrease the possibility of being trapped into the local optimum, the new PSO introduces genetic selection strategy: To set particle number in the swarm as *m*, father population and son population add up to *2m*. To select *q* pairs from m randomly from individual particle *i*. If the fitness value of *i* is smaller than its opponents, *i* will win out and then add one to its mark and finally select those particles which have the maximum mark value into the next generation. The experiments conducted show that this strategy greatly reduces the possibility of being trapped into a local optimum when solving certain functions.

VII. EXPERIMENTAL RESULTS AND ANALYSIS

1) *EXPERIMENTAL SETUP*

This research work is experimented and analyzed with the cloudsim used and it consists of 10 resources with different processing speed. Due to the nature of metaheuristic algorithm and random initial positions, each algorithm has been implemented 30 times on average and the obtained average results are considered as a final answer and criteria for comparison.

The minimum computation cost value of the best solutions is recorded throughout the optimization of 50 iterations of all tasks completed. The test has been conducted for the task scheduling problem from 10 processors with 100 tasks. In IPSO algorithm, the parameters were set such that the number of particle is 50, the self-recognition coefficient *c1* and social coefficient *c2* are 2 and the weight w is 0.9. The experimental parameter settings of PSO and IPSO algorithms are shown in Table 3.1.

Table 4.1 Parameters and its value for PSO and IPSO

Parameter description	**Parameter value**
Size of Swarm	50
Self-recognition coefficient *c1*	2
Social coefficient *c2*	2
Weight *w*	0.9
Iterations	50

2) EXPERIMENTAL RESULTS

Fig.7.1-7.5 plots the convergence of total cost computed by PSO and IPSO over the 50 number of iterations for different sizes of total data processed by the workflow such as 64 MB, 128 MB, 256 MB and 512 MB respectively. Initially, the particles are randomly initialized. Therefore, the initial total cost is always high. This initial cost corresponds to the 0th iteration. As the algorithm progresses, the convergence is drastic and it finds a global minima very quickly. The average number of iterations needed for the convergence is seen to be 30-35, for this application environment. It displays that IPSO usually had better average completion time values than that of PSO.

Fig.7.1 shows the computation cost of PSO and IPSO scheduling algorithm for 64 MB. For IPSO, the number of iterations needed for the convergence is seen to be 40. It shows IPSO usually spent the shorter time to complete the scheduling than PSO algorithm. It is to be noted that IPSO usually spent the shorter time to accomplish the various scheduling tasks and had the better result compared with PSO algorithm.

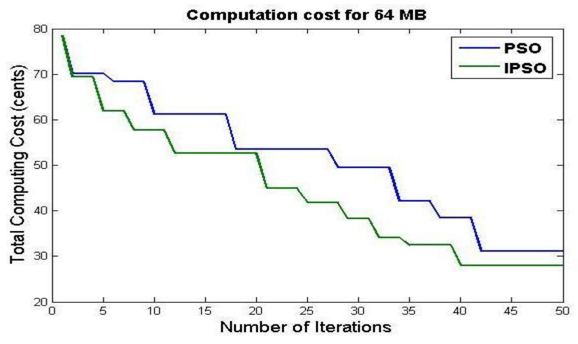

Figure 7.1 Performance of PSO and IPSO scheduling algorithm for 64 MB

Fig.7.2 shows the computation cost of PSO and IPSO scheduling algorithm for 128 MB. For IPSO, the number of iterations needed for the convergence is seen to be 36.

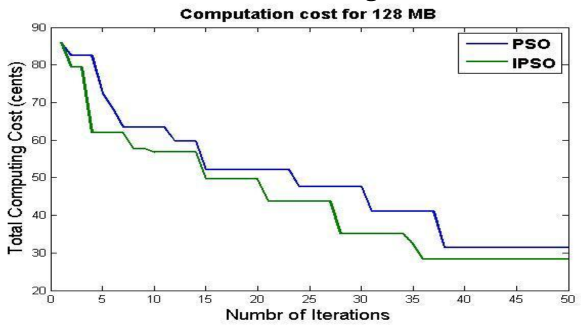

Figure 7.2 Performance of PSO and IPSO scheduling algorithm for 128 MB

Fig.7.3 shows the computation cost of PSO and IPSO scheduling algorithm for 256 MB. For IPSO, the number of iterations needed for the convergence is seen to be 40.

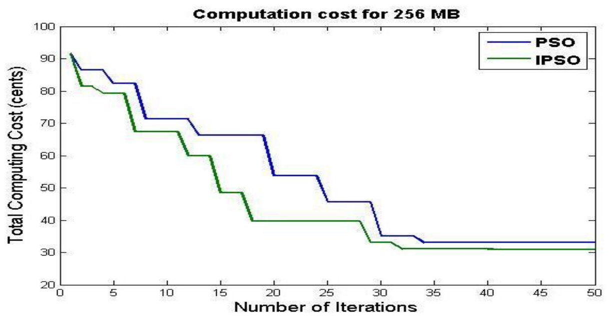

Figure 7.3 Performance of PSO and IPSO scheduling algorithm for 256 MB

Table 4.2 plots comparison of optimal total cost between PSO based resource selection and IPSO algorithms when varying total data size of a workflow. IPSO achieves 10.19 percentages of improvements for 64 MB of total data processed than the PSO algorithm. For 128 MB and 512 MB, the proposed IPSO method attains 9.89 and 6.83 percentage of improvements respectively. For 1024 MB the proposed IPSO method returns 4.83 percentage of improvements in optimal total computation cost. Clearly, IPSO based mapping has much lower cost as compared to that of the existing PSO based mapping. In addition, the slope of the trend line of all the figures shows that PSO based mapping reduces the cost linearly, whereas the IPSO reduces exponentially and maintains a balanced the intensification and diversification in the entire search space.

Table 4.2 Comparison of optimal minimum cost of computation with various data size for PSO and IPSO

Size of Data	PSO	IPSO	Percentage of Improvement
64 MB	31.19	28.01	10.19%
128 MB	31.32	28.22	9.89%
256 MB	33.03	30.96	6.26%

VIII. CONCLUSION

The IPSO algorithm for workflow scheduling is capable of overcoming the poor convergence problem of PSO method. It focuses on mapping task and resource with minimum computation cost. In the IPSO method the worst positions of the individual particle are recorded and applied in the whole swarm. Thus it may make individual particles move in the direction of evading the worst positions of the individual particle and of the whole flock. This will surely enlarge the global searching space of particles and enable them to avoid being trapped into a local optimum too early. The result of total cost of execution was obtained by varying the data size and is plotted in various figures and also comparison is made with IPSO against PSO. It is found that IPSO based task-resource mapping can achieve better cost savings when compared to PSO based mapping for application workflow.

REFERENCES

[1] Bittencourt, LF, Sakellariou, R & Madeira, RM, 2010, 'DAG Scheduling Using a Look ahead Variant of the Heterogeneous Earliest Finish Time Algorithm', Proceedings of 2010 18th Euromicro conference on Parallel, Distributed and Network-based Processing, pp. 27-34.

[2] Blum, C & Roli, A, 2003, 'Metaheuristics in combinatorial optimization: Overview and conceptual comparison', Journal of ACM Computing Surveys, vol. 35, no. 3, pp. 268-308.

[3] Chen, WN & Zhang, J, 2009, 'An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements', IEEE Transactions on Systems, Man, and Cybernetics, vol. 39, no. 1, pp. 29–43.

[4] Deneubourg, JL & Goss, S, 1989, 'Collective patterns and decision making', Ethology, Ecology and Evolution, vol. 1, no. 4, pp. 295–311.

[5] Deneubourg, JL, Goss, S, Franks, N, Franks, AS, Detrain, C & Chretien, L, 1990, 'The dynamics of collective sorting robot-like ants and ant-like robots', Proceedings of the first international conference on simulation of adaptive behavior on from animals to animats, pp. 356–363.

[6] Eberhart, R & Kenedy, J, 1995, 'Particle swarm optimization', Proceedings of IEEE International Conference on Neural Networks, pp. 1114–1121.

[7] Fesanghary, M, Damangir, E & Soleimani, I, 2009, 'Design optimization of shell and tube heat exchangers using global sensitivity analysis and harmony search algorithm', Applied Thermal Engineering, vol. 29, no. 6, pp. 1026–1031.

[8] Sadhasivam,N & Thangaraj,2017,' Design of an improved PSO algorithm for workflow scheduling in cloud computing environment', Intelligent Automation & Soft computing,vol.23,no.3,pp.2017

[9] Glover, F, 1986, 'Future Paths for Integer Programming and Links to Artificial Intelligence', Computers and Operations Research, vol. 13, no. 5, pp. 533-549.

[10] Guo, L, Zhao, S, Shen, S & Jiang, C, 2012, 'Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm', Journal of Networks, vol. 7, no. 3, pp. 547-553.

[11] He, HD, Lu, WZ & Xue, Y, 2014, 'Prediction of particulate matter at street level using artificial neural networks coupling with chaotic particle swarm optimization algorithm', Building and Environment, vol. 78, no. 8, pp. 111–117.

[12] Huang, J, Wu, K, Leong, LK, Ma, S & Moh, M, 2013, ' A tunable workflow scheduling algorithm based on particle swarm optimization for cloud computing', The International Journal of Soft Computing and Software Engineering, vol. 3, no. 3, pp. 351-358.

[13] Jau, YM, Su ,KL, Wu ,CJ & Jeng, JT, 2013, 'Modified quantum-behaved particle swarm optimization for parameters estimation of generalized nonlinear multi-regressions model based on Choquet integral with outliers', Applied Mathematics and Computation, vol. 221, no. 9, pp. 282–295.

[14] Karger, D, Stein, C & Wein, J, 2010, 'Scheduling Algorithms. Algorithms and theory of computation handbook', Special topics and techniques, CRC Press, Florida, United States.

[15] Kashan, AH & Karimi, B, 2009, 'A discrete particle swarm optimization for scheduling parallel machines', Computers and Industrial Engineering, vol. 59, no. 1, pp. 216-223.

[16] Kaveh, A & Talatahari, S, 2009, 'Engineering Optimization with Hybrid Particle Swarm and Ant Colony Optimization', Asian Journal of Civil Engineering Building and Housing, vol. 10, no. 6, pp. 611-628.

[17] Li ,CS, Zhou, J, Kou, P & Xiao J, 2012, 'A novel chaotic particle swarm optimization based fuzzy clustering algorithm', Neuro computing, vol. 83, no. 4, pp. 98–109.

[18] Li, P & Xiao, H, 2014, 'An improved quantum-behaved particle swarm optimization algorithm', Applied Intelligence, vol. 40, no. 3, pp. 479–496.

[19] Liu, B, Wang, L & Jin, YH, 2008, 'An effective hybrid PSO-based algorithm for flow, shop scheduling with limited buffers', Computers and Operations Research, vol. 35, no. 9, pp. 2791-2806.

[20] Liu, Z, & Wang X, 2012, 'A PSO-based algorithm for load balancing in virtual machines of cloud computing environment', Proceedings of the Third international conference on Advances in Swarm Intelligence, pp. 142-147.

[21] Maguluri, ST & Srikant, R, 2014, 'Scheduling jobs with unknown duration in clouds', IEEE/ACM Transactions on Networking, vol. 22, no. 6, pp. 1938–1951.

[22] Moraga, R, DePuy, G & Whitehouse, G, 2006, Metaheuristics: A solution methodology for optimization problems: Handbook of Industrial and Systems Engineering, CRC Press, Florida, United States.

[23] Pandey, S, Wu, L, Guru, SM, & Buyya, R, 2010, 'A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments', Proceedings of 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 400-40.

[24] Qin, X, Yang, Z, Li, W & Yang, Y, 2013, 'Optimized task scheduling and resource allocation in cloud computing using PSO based fitness function', Information Technology Journal, vol. 12, no. 23, pp. 7090–7095.

[25] Rodrigues, D, Pereira, L, Almeida, T, Papa, J, Souza, A, Ramos, C & Yang, XS, 2013, 'BCS: A binary cuckoo search algorithm for feature selection', Proceeding of IEEE International Symposium on Circuits and Systems, pp. 465–468.

[26] Tasgetiren, MF, Liang, YC, Sevkli, M & Gencyilmaz, G, 2007, 'A particle swarm optimization algorithm for make span and total flow time minimization in the permutation flowshop sequencing problem', European Journal of Operational Research, vol. 177, no. 3, pp.1930-1947.

[27] Torabi, SA, Sahebjamnia, N, Mansouri, SA & Bajestani, MA, 2013, 'A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem', Applied Soft Computing, vol. 13, no. 12, pp. 4750-4762.

[28] Tsai, WC, Huang, WC, Chiang, MH, Chiang, MC & Yang, CS, 2014, 'A hyper-heuristic scheduling algorithm for cloud', IEEE Transactions on Cloud Computing, vol. 2, no. 2, pp. 236–250.

[29] Tseng, CT & Liao, CJ, 2008, 'A particle swarm optimization algorithm for hybrid flow- shop scheduling with multiprocessor tasks', International Journal of Production Research, vol. 46, no. 17, pp. 4655-4670.

[30] Voss, S, 2001, Meta-heuristics: The state of the art: Local Search for Planning and Scheduling, Lecture Notes on Artificial Intelligence, 2148, pp. 1-23.

[31] Zeng, YJ & Sun, YG, 2014, 'An improved particle swarm optimization for the combined heat and power dynamic economic dispatch problem', Electric Power Components and Systems, vol. 42, no. 15, pp. 1700–1716.

[32] Pandey, S., Wu, L., Guru, S.M., & Buyya, R.,2010, ' A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments', In Advanced Information Networking and Applications ,24th IEEE International Conference, pp.400-407.

[33] Zhang, H, Fernández, JA, Rangaiah GP, Petriciolet, AB & Segovia, JG ,2011, 'Evaluation of integrated differential evolution and unified bare-bones particle swarm optimization for phase equilibrium and stability problems', Fluid Phase Equilibria, vol. 310, no. 2, pp. 129–141.

[34] Zhu, X, Yang, LT, Chen, H, Wang, J, Yin, S & Liu, X, 2014, 'Realtime tasks oriented energy-aware scheduling in virtualized clouds', IEEE Transactions on Cloud Computing, vol. 2, no. 2, pp. 168–180.

[35] Zhu, Z, Chen, C, Yang, LT & Xiang, Y, 2015, 'ANGEL: agent based scheduling for real-time tasks in virtualized clouds', IEEE Transactions on Computers, vol. 64, no. 12, pp. 3389–340.

[36] Zhu, Z, Zhang, G, Li, M & Liu, X, 2016, 'Evolutionary Multi-objective workflow scheduling in cloud', IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 5, pp. 1344–1357.

[37] Zuo, X, Zhang, G & Tan, W, 2014, '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.