ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

REVIEW PAPER

Available Online at www.ijarcs.info

Workflow Scheduling in Cloud Using Nature Inspired Optimization Algorithms

Palvi Mahajan
Department of Computer Engineering and Technology
Guru Nanak Dev University
Amritsar, Punjab, India

Keshav Dhir
Department of Computer Engineering and Technology
Guru Nanak Dev University
Amritsar, Punjab, India

Amit Chhabra
Department of Computer Engineering and Technology
Guru Nanak Dev University
Amritsar, Punjab, India

Abstract: Cloud computing is one of the fastest growing technologies in the world. In the cloud computing scenario, selecting and assigning resource from a large pool of resources to workflow tasks is difficult and a well-known NP-hard problem. Metaheuristics are used to provide near optimal solution to resource assignment problem in cloud and distributed computing systems. In this paper we have presented a survey of variety of meta-heuristic approaches such as genetic algorithm, particle swarm optimization, ant colony optimization, Cat Swarm, Bat, firefly and Grey Wolf Optimizer algorithms which are used for workflow scheduling in cloud environment.

Keywords: Meta-heuristics, Cat Swarm, Firefly Algorithm, Bat Algorithm and Grey Wolf Optimization algorithm.

I. INTRODUCTION

Cloud computing has become an essential part of the computing environment and its use is inevitable. Defined by NIST[1] "As a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". A scheduling mechanism is required to map resources to tasks from large pool of resources.

In general, scheduling (or mapping) is a kind of NP-hard problem. It becomes even more difficult in case of workflow tasks consisting of interdependent tasks that are bounded together in a parent child relationship. Thus, it is not advisable to have a comprehensive solution for these kind of problems where it becomes typically impossible to get the seamless deterministic solution within a given time span. Meta-heuristics techniques can find near-optimal solutions in reasonably shorter interval. Metaheuristics refer to a demanding procedure which aims at discovery, producing or choosing the right heuristic that can provide an approximate solution to an optimization problem [3]. In this paper we have focused on some of the nature based meta-heuristics algorithms for tackling the problem of scheduling a workflow.

A. Workflows

A workflow generally can be described as group of task to be executed in a sequence to obtain a particular result. The tasks have data dependencies which gives it a parent child kind of a relation, where a child cannot be executed until and unless its parent has been executed [10]. This dependency among them can be represented via a directed acyclic graph (DAG). The graph G (v,e) represents a DAG where 'v' show the nodes and 'e' represents the task dependencies among them. We can depict a workflow as in the following diagram:

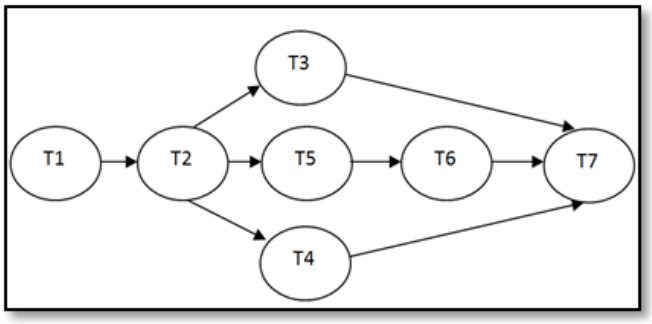

Figure 1. Graph representation of a workflow.

In Figure 1 we have T1, T2, T3, T4, T5, T6, T7 which represent the various task along with its dependencies. Task T2 has T1 as its parent task, similarly T5, T4, T3 can be executed after T2, T6 after T5 and T7 will have to wait for all the tasks to executed and act as the exit or the completion node.

Scheduling a workflow in the context of cloud means to allocate the resource to the tasks so as to meet performance objective criterion such as minimized makespan, minimized minimum mean flow time or deadline constraint etc. Some of the major performance objectives are:

- Budget: It is the pre-decided cost that a customer pays for the usage of the cloud resources.
- Reliability: Is the mean time of a resource to recover from a failure.
- Throughput: Number of workflow tasks executed in a given time.
- Makespan: It is the time when the last task of the workflow is executed

- Load balancing: A scheduler needs to optimize the resources for proper utilization and avoid overloading
- Cost: We need to minimize the cost can be of distributing loads to virtual machine, having resources etc.

For scheduling generally two types of algorithm static algorithm and dynamic algorithm [13] are available in literature; in static algorithm the estimation of execution time, its structure and resources is already known before the execution but in dynamic this information is available at job ready state before execution. Since scheduling of workflows in cloud environment is known to be a NP hard problem, therefore various nature-based optimization algorithms are used for this purpose.

II. RELATED WORK

Various meta-heuristics have been applied in the field of workflow scheduling. Majority of these are based on certain traits and behaviour of living organisms, molecular, swarm of insects, and nerve-based systems like particle swarm optimization (PSO), Genetic algorithm (GA), Ant colony optimization (ACO).

A. Particle Swarm Optimization

It is based on the behavior of a huge number also known as colony or swarm of insects, such as ants, termites, bees and a flock or group of birds or a pool of fish. The particle swarm optimization algorithm depicts the behavior of these communal organisms. Optimization methods based on swarm intelligence are called behaviorally enthused algorithms as opposed to the innate algorithms, which are called evolution-based techniques. A population based stochastic optimization algorithm developed by Kennedy and Eberhart in 1995 [21, 22]. In this type of workflow based scheduling, the particle dimensions represent the number of tasks whereas the mapping between Virtual Machines and tasks is done by their positions. Each particle has its own position in space and corresponds to a candidate solution. Sometimes it creates an arbitrary swarm of particles but its better to have a good initial swarm to produce better results [23, 24].

In 2010, A PSO based workflow scheduling algorithm accounting both the cost of computation and data transmission was presented which computes workflow application mapping of total cost [25]. It has a re-computation mechanism which makes it reliable and dynamically balance while task mapping in order to get all the ready workflow tasks well scheduled.

In 2012, workflow scheduling scheme to minimize the cost and makespan and also to increase the QoS addressing reliability, was proposed known as Set Based PSO (S-PSO) [27]. It is appropriate for cloud workflow scheduling problem. It consists of several QoS constraints and various heuristics to get the fitness function criteria for scheduling.

In 2013, in order to get a minimum cost to makespan, a tunable function model to map the tasks was presented by Huang and further to address the main problem for a smaller makespan. It first calculates the fitness value after random initialization of particles and then further to test the overall best position among the particles to update their velocity and position. This all happens while the stopping criterion is not met to achieve of optimal mapping.

In 2014, three different PSC based workforce scheduling algorithms work proposed, namely - BPSO, Discrete PSO and Local Minima Jump PSO. In BPSO [24], main target is to reduce the implementation cost and time alongside considering both the budget and deadline constraints. This is a priority based workflow task execution scheme. In Discrete PSO [28], the main target is to reduce the finishing time and cost while taking into account the security issues. It consists of three main stages or levels: initialization, execution and the end. To get the feasible solution in execution phase, QoS values are calculated to achieve the optimal scheduling solution at the end. The Local Minima PSO [29], which does the load balancing alongside producing of cost and time to get minimum makespan. Velocity of each particle is increased when poorer convergence is found while iterating.

In 2015, many improved PSO initialization particles based algorithms were found which further helped in better workflow scheduling [23].

B. Genetic Algorithm

It is based on the principles of natural inheritances and natural selection. Genetic algorithms (GAs) are well suited for solving problems like mixed continuous–discrete variables, and discontinuous and non-convex design spaces. GAs has shown to provide the global ideal solution with a high possibility. Although Holland first presented GAs, the basic ideas of scrutiny and design based on the conceptions of biological progression can be found in the Rechenberg's work. Basic elements of natural inheritances—imitation, crossover, and mutation—are used in the genetic type search procedures. At first the population was generated at random, but the concept of better initial population is proposed to achieve better results [30, 31].

A different GA based on bi-objective scheduling outline called DWSGA [32] is suggested which is based on two main resource factors: the completion time and workload distribution. In order to find the better solution and also consuming lesser time, it initially makes a good population. Then in order to get the most appropriate solution, it optimizes the makespan and workload distribution on available resources.

In 2011, GA which uses MDP i.e., Markov's Decision Process [33], is capable of observing tuning schedule and temporal dynamics was proposed. In this paper, the user first submits the QoS constrained workflow in order to calculate the average execution time to run the given task. The execution module performs the scheduling plan which returns the result to the user interface. The real cost and makespan acquired is reimbursed by the QoS monitor after the completion of provisioning schedule on the cloud.

In 2013, BCHGA – a Hybrid GA [34], in order to optimize the performance and data transfer cost with a budget limitation was presented. It creates two levels: Top and Bottom of population, which are first calculated to create the initial population. Then it assigns them to available virtual machines. While the termination criteria are not met, it evaluates the fitness function of each individual in the list of population. The selection operator selects the parent, then the crossover type operator is applied on the carefully chosen parent to create child and then further to apply mutation operator on that child to get a valid child which is further added to new population.

C. Ant Colony Optimization

It is based on the supportive conduct of real ant clusters, which are able to find the direct & convenient path from their home to a source of nourishment. Dorigo established this method with his associates in the early 1990s. In order to meet the sub deadlines and to optimize the execution cost of workflow based tasks which were left incomplete at private clouds to meet their requirements needed to migrate to public clouds an ACO based algorithm was proposed [36, 37]. ACO helps to assign the cost effective VMs to those missed or uncompleted tasks that were migrated before, which can further be able to execute the workflow applications within sub deadlines.

In 2012, an ACO based algorithm known as MPA i.e., Multiple Pheromone Algorithm [39] was proposed to palliate the task completion time, resource utilization and makespan by using different pheromone values for selective tasks having different resources. The pheromone value keeps on updating after each task completion by the ants. This ACO operates based on the makespan, cost and reliability constraints.

In 2013, a solution based on ACO was proposed, which consists of ants divided into two main groups: Forward-Ants and Backward-Ants [38]. In this, the forward ants are used to locate and find the accessible VMs for producing Backward-Ants, when they themselves find the computation resources. It consists of the Master Node which is used to receive the jobs in batches. The Backward-Ants leaves pheromones to produce further Backward-Ants which can locate the valuable resources. The pheromone keeps on updating beside the execution of the jobs. The Master Node assigns the tasks left uncompleted to other nodes until all the jobs are executed.

III. CAT SWARM OPTIMIZATION

CSO was introduced in 2007 by Tsai and Chu, which is a swarm based evolutionary algorithm based on the seeking and tracing behavior of cats:

- Seeking mode: This behavior is exhibited by cats when they stay in one position and calculate the next best move.
- Tracing mode: While chasing the target they move with high velocity towards their next position.
- MR (mix ratio) which comprises of cat from the population of both tracing and seeking mode.

We can implement the CSO by following the steps shown in figure 2:

A. Application in Workflow scheduling

In 2014, Bilgaiyan introduced a CSO [9] algorithm based scheduling technique which considers both the cost of transmitting data between the two dependent resources and execution cost of task on different resources. This works in two stages seeking mode and tracing mode by using them the energy wastage in random movements is reduced.

1. The swarm is initialized with 'C' cats in 'D' dimensions
2. Assign V_{jd} velocity in random order to the j^{th} cat having a position X_{jd} here 'd ' represent the index of the dimension
3. We randomly assign a mix ratio(MR) from the seeking mode to the tracing mode
4. The fitness of each cat is assessed and location of best cat is stored in memory
5. Depending on the mode they are in, we update the position of the cats
6. Terminate if criteria is met, else goto 3.

Figure 2. CSO Algorithm

The tasks and resources are here indicated by the cats and their mapping depends on the mode that cats are in. Fitness value of the cat is assessed which helps to map with the minimum cost thus the final solution obtained is the best mapping with the minimum cost. Bilgaiyan put forward a MOCSO [12] which can be used to address the multi objective scheduling problems. A pareto-front graph is made by assigning the workflows as the dimensions and the cats as the schedule between the resources available and the sets of the task. The main aim of using this scheme was to decrease the CPU idle time, make-span and the execution cost. By applying MOCSO it was seen that it gives comparatively large number of solutions within a fixed number of iteration than other algorithms.

IV. BAT ALGORITHM

The Bat Algorithm is said to be inspired from the echo-location behavior of bats. This ability of echo-location of micro-bats is engrossing as these bats are able to search their prey and distinguish between various types of insects even in complete unenlightened state. These bats give off a very high sound pulse and listen to the echo that bounces back from the surrounding objects. Their pulses can differ in properties and can be strongly related with their hunting strategies, depending upon their species. Their signal bandwidth varies.

Bat algorithm is a meta-heuristic algorithm which is based on three approximate or idealized rules:

- The echo-location is used as an evaluation technique by bats to evaluate the distance of their food/prey. Also food and environmental obstruction can be well differentiated by them.
- Bats tend to fly with certain amount of velocity towards a specific position where velocity and position are represented as v_i and x_i respectively. Among frequency and wavelength, one of the factors is kept fixed and the other is automatically adjusted by bats and parameters named as pulse rate and loudness are adjusted based on the distance from the target.
- Usually loudness can vary from a large (positive) to a lowest possible constant value [13].

1. **Objective function** $f(Y) = Y(y_1, y_2 \ldots y_n)^T$
2. Bat population is initialized Y_i *(i=1, 2…x)*
3. v_i Defines frequency of pulse f_i at Y_i
4. Initialize the pulse rate as r_i and loudness as L_i
5. **while** (I < Max numbers of iterations)
6. Calculate new solutions by changing the frequency,
7. And updating velocities, locations and solutions
8. **if** (*random* $> r_i$)
9. Select a solution among the best solutions
10. Generate a local solution around the selected best solution
11. **end if**
12. Generate a new solution by flying randomly
13. **if**(*random* $< L_i$ & $f(y_i) < f(y^*)$)
14. Accept the new solution, that increases r_i and reduces L_i
15. **end if**
16. Rank the bats and find the current best y^*

Figure 3. Bat Algorithm

1. **Objective function** f(Y), where Y = $(y_1, y_2, \ldots, y_n)$
2. The initial population of x fireflies is generated, Y_i,
3. Here i =1, 2,…..,x
4. Intensity of light I_i at Y_i is calculated by f(Y_i)
5. Light absorption co-efficient is defined as γ
6. **while** (t <MaxGen)
7. **for** = 1 : x , all f fireflies
8. **for** j = 1 : x, all f fireflies (inner loop)
9. **if** ($I_i < I_j$), Move fireflies *i* towards *j*;
10. **end if**
11. Attractiveness varies with distance *i* via exp [$-\gamma r^2$]
12. **end** for j
13. **end** for i
14. The fireflies are ranked and best solution is calculated
15. **end while**
16. Process-post the results

Figure 4. Firefly Algorithm

A. Application in Workflow Scheduling

Binary bat algorithm was developed by Nakamura [14] the only difference between the binary bat and bat algorithm was it takes discrete inputs of binary digits. It was used by Raghavan and Marimuthu [15] for scheduling workflows in cloud environment working with the workflow can be difficult because of the interdependent nature of the tasks but binary bat algorithm proves to be better than previous algorithm of its type on basis of execution cost of each task as well as makespan of the workload. Only one task can be assigned to each resource. Afterwards a multi-objective scheduling problem was solved by bat algorithm [16] to minimize the execution time, maximize the reliability keeping in mind that the cost does not increase the budget assigned by the user. This algorithm proves to work better than comparison other algorithms.

V. FIREFLY ALGORITHM

Firefly algorithm is inspired on the swarm behavior of fireflies which usually known to exist in groups. The fireflies attract with each other on the basis of their blinking light attribute which is also used to defend themselves from other predators. The swarm of fireflies usually moves towards the direction of the brightest one. Fireflies with lower light intensities move toward the ones with higher light intensities. Light intensity of fireflies increases with the increase in the distance between the fireflies [5]. Following properties of fireflies were taken in account for developing Firefly algorithm:

- All the fireflies will be interested in each other due to their unisex attribute.
- Among fireflies their attractiveness is equivalently related to their lighting, and among any of two fireflies, the less lustrous one will be pulled or attracted by the more lustrous one. However the brightness decrease as their distance increases.
- Feature of brightness among the fireflies is calculated as the landscape of the objective function.

A. Application in Workflow Scheduling

The firefly algorithm was used in 2014, by A.Paulin Florence and V.Shanthi [40] to achieve load balancing with maximized memory and CPU utilization. In 2016 the firefly algorithm was used to handle the workflow scheduling by R. Sundar Rajan, V. Vasudevan, and S. Mithya [7]. Fireflies represent virtual machines thus by calculating the brightness and distance factor the best virtual machine is selected to execute the job. Thus the study shows that firefly algorithm can decrease the execution time in workflow scheduling thus decreasing the cost. This was found to be the best among swarm optimization algorithms for complicated large scale cloud environments. It distributes the loads evenly and along with it reduces the overall make-span.

VI. GREY WOLF OPTIMIZER

Inspired by the hunting and living behavior of the wolves GWO algorithm was introduced by Mirjalili in 2014. The wolves use hierarchy structure in their group.

- Highest level: Alpha (α) wolf is the leader of the group which manages the group and is the only one allowed to mate
- Second level: Beta (β) wolves are the connection between the lower levels and the alpha wolf.
- Third level: Delta (δ) wolves, they subordinate the upper levels by guarding the group, hunting etc.
- Lowest level: Omega(Ω) wolves, just obey the high level wolves.

W(t) and $W_p(t)$ are taken as position of the grey wolf and position of the prey respectively. Where t is the given iteration, A and C are vectors A= $2b.r_1$-b and C= $2r_2$ here r_1, r_2 represent the range of random number between [0, 1] and b gradually decreases from 2 to 0. The position of alpha, beta and delta for hunting is updated by:

$D_\alpha = |C_1 . W_\alpha(t) - W(t)|$, $D_\beta = |C_2 . W_\beta(t) - W(t)|$, $D_\delta = |C_3 . W_\delta(t) - W(t)|$

$W_1 = W_\alpha(t) - A_1.(D_\alpha)$, $W_2 = W_\beta(t) - A_1.(D_\beta)$, $W_3 = W_\delta(t) - A_1.(D_\delta)$

$W(t+1) = W_1 + W_2 + W_3/3$

1. Initialize the population of the grey wolves W_z(z =1,2,3….n) and value of A, C and b
2. Calculate the fitness value for each wolf
3. Find alpha(W_α), beta(W_β) and delta(W_δ)
4. While t < n
5. Value of each wolf is updated W_z(z =1,2,3….n) based on the equation $W(t+1) = W_1+W_2+W_3/3$
6. Calculate the fitness of all the population
7. Update W_α , W_β , W_δ
8. t = t+1
9. end while
10. Return W_α

Figure 5. Grey Wolf Optimization algorithm

A. Application in Workflow Scheduling

Wolf optimizer algorithm is made to work in the continuous spaces and the task resource mapping is a discrete problem so it cannot be directly applied for mapping the task and resources. To implement workflow scheduling using the GWO algorithm, Khalili and Sayede introduced a Pareto based grey wolf optimizer (PGWO). In which they use a smallest position value (SPV) which maps the continuous position vector value to discrete values X_i: $<x_{i1}, x_{i2}, x_{i3}, ..x_{in}>$ and the permutation vector Y_i: $<y_{i1}, y_{i2}, y_{i3}, ..y_{in}>$ is obtained for the i^{th} wolf using the module $y_{i,k}$= mod($x_{i,k}$, m). In simple words we can say that for each member of vector X_i, like for $X_{i,j}$ shows the virtual machine on which j^{th} should be executed. This algorithm was used for the solution of multi objective problems by using pare to fronts, for them to be idle they need to be accurate, well distributed and well spread.

Table I. Summarization of Algorithms

ALGORITHM	MODIFIED VERSIONS	REFERENCES	AUTHOR	OBJECTIVES FULFILLED	TOOLS USED
PARTICLE SWARM OPTIMIZATION	Standard PSO	[21, 22]	Nallakumar R., Rahman M	Stochastic Optimization	-
	S-PSO	[27]	Chen W-N, Zhang J	Minimization of Cost, Makespan, QoS increment	Amazon EC2
	BPSO	[24]	Verma A, Kaushal S	Reducing Cost and execution Time	Java
	Discrete PSO	[28]	Jianfang C., Junjie C., Qingshan Z.	Reducing Cost while considering Security	Cloud Sim
	LMJ-PSO	[29]	Chitra S.	Load Balancing for minimum makespan	-
GENETIC ALGORITHM	DWSGA	[32]	Aryan Y, Delavar AG	Completion time, Workload distribution	-
	MDP-Based	[33]	Barrett E, Howley E, Duggan J.	Scheduled tuning, Qos Constraints	Cloud Sim
	BCHGA-Hybrid GA	[34]	Verma A,Kaushal S.	Optimize Execution & Data transfer Cost	Java
ANT COLONY OPTIMIZATION	Standard ACO	[36, 37]	Singh L, Singh S, Singh R	Optimize Cost and Workflow tasks	Cloud Sim
	MPA	[39]	Gogulan R, Kavitha A, Karthick Kumar	Task Completion and Resource Utilization	Java
	Extended ACO	[38]	Zhou Y, Huang X	Finding Available Resources and Utilization	Cloud Sim
CAT SWARM OPTIMIZATION	CSO	[9]	Bilgaiyan	Load distribution, Minimizing cost	Workflow Sim
	MOCSO	[12]	Bilgaiyan	Minimizing makespan, Minimizing cost, CPU idle time	MATLAB
BAT ALGORITHM	BBA	[15]	S Raghavan, Marimuthu C, Sarwesh P.	Minimizing cost	-
	MOBA	[16]	Navneet Kaur, Sarbjeet Singh	Minimizing makespan, Reliability, Budget constraint, Minimizing cost	Workflow Sim
FIREFLY ALGORITHM	MOFA	[40]	A.Paulin Florence, V.Shanthi	CPU utility rate, Memory usage rate, load balancing	Cloud Sim
	MOFA	[17]	SundarRajan, V. Vasudevan, S. Mithya	Minimizing makespan, Load distribution, Minimizing cost	Cloud Sim
GREY WOLF OPTIMIZER	PGWO	[20]	Azadekhalili, Seyedbabamir	Minimizing makespan, Throughput, Minimizing cost	Cloud Sim

It minimizes make-span along with the cost of the virtual machines and also maximizes the throughput of resources. Future scope encounters that multi-objective PGWO need to work more while has to work in balanced workflows that have less distance from each other.

VII. CONCLUSION

In this paper various nature based optimization algorithm have been studied which are used to schedule workflow tasks. During comparative analysis shown in table 1, we have seen how nature based algorithms such as firefly algorithm, cat swarm optimization; bat algorithm and grey wolf optimizer algorithm are used to find optimal schedule for workflows applications. The potential of cloud computing can be applied in diverse applications in different areas using efficient scheduling.

VIII. REFERENCES

[1] "Taxamony on workflow management systems for grid computing" by R. Buya and J.Yu.

[2] "Effective and fast task scheduling in heterogeneous system" by A. Gemund, A. Redulescu in HCS 2000.

[3] "Critical-Path dynamic scheduling: An effective way to allocate task graphs to multiprocessor" by I. Ahmad, Y. Kwok in IEEE may 1996.

[4] "NIST definition of cloud computing(draft)" by Grance, peter and mell in special NIST publication 2011.

[5] "Workflow scheduling technique to consider task processing rate in spot-instance based cloud" by H.Yu, j.lim in Frontier and innovation in computing and communication.

[6] "Metaheuristics: From design to implementation" by El Ghazalitalbi, 2009.

[7] "Computational intelligence based on the behavior of cats" by Pei-Wei, Chu. International journal of innovative computing, information and control, February 2007.

[8] "Cat swarm optimization" by S.C.Chu, P.W.Tsai and J.S.Pan in Springer 2006.

[9] "Workflow scheduling in cloud computing using cat swarm optimization" by Bilgyan, Santwana in IEEE 2014.

[10] "Survey and future trend of study on multi-objective scheduling" by Wang, Gao and Zangin 4th IEEE conference 2008.

[11] "Solving multi-objective problems using cat swarm optimization" by P. M. Panda, Pardhan in expert system application 2011.

[12] "Multi-objective CSO algorithm for workflow scheduling in cloud enviroment" by Bilgyan, Santwana, das in 2015 springer.

[13] "New metaheuristic bat-inspired algorithm" by X.S Yang in NICSO springer 2010.

[14] "BBA- binary bat algorithm for feature selection" by Nakamura, Costa, Pereira in 25th conference SIBGRAPI 2012.

[15] "Bat algorithm for scheduling workflow applications in cloud" by S Raghavan, Marimuthu C, Sarwesh P in EDCAV 2015.

[16] "Budget-Constrained Time and Reliability Optimization BAT Algorithm for Scheduling Workflow Applications in Clouds" by Navneet Kaur, Sarbjeet Singh in EUSPN 2016.

[17] "Scheduling in Cloud Computing Environment using Firefly Algorithm" by SundarRajan, V. Vasudevan, S. Mithyain ICEEOT, 2016.

[18] "Firefly algorithm for noisy non-linear optimization problem" by P.Aungkulanon, N.Chai.ead in national inter-conference of engineers and computer scientists, 2011.

[19] "Evolutionary multi-objective optimization tutorial" by Bleuler, Zitler in 2004 springer.

[20] "Optimal scheduling workflows in cloud computing environment using pareto-based grey wolf optimizer " by Azadekhalili, Seyedbabamir in 2016 wiley.

[21] "A survey on deadline constrained workflow scheduling algorithms in cloud environment" by Nallakumar R. arXìv preprint arXìv:1409, 7916; 2014.

[22] "Adaptive workflow scheduling for dynamic grid and cloud computing environment" by Rahman M, et al. Concurr Comput: PractExp 2013;25(13):1816-42.

[23] "Cost minimized PSO based workflow scheduling plan for cloud computing" by Verma A, Kaushal S; 2015.

[24] "Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud" by Verma A, Kaushal S. In: Proceedings of the 2014 Recent Advances in Engineering and Computational Sciences (RAECS); 2014. IEEE.

[25] "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments" by Pandey S, et al. In: Proceedings of the 2010 24th IEEE International Conference on advanced information networking and applications (AINA) 2010. IEEE.

[26] "A tunable workflow scheduling algorithm based on particle swarm optimization for cloud computing" by Huang J, et al. Criterion 2013;12:14.

[27] "A set-based dìscrete PSO for cloud workflow scheduling with user-defined QoS constraints" by Chen W-N, Zhang J. In: Proceedings of the 2012 IEEE international conference on systems, man, and cybernetics (SMC).

[28] "An optimized scheduling algorithm on a cloud workflow using a discrete particle swarm" by Jianfang C, Junjie C, Qingshan Z. Cybern Inform Technol 2014;14 (1): 25—39.

[29] "Local minima jump PSO for workflow scheduling in cloud computing environments" by Chitra S, et al. In: Advances in computer science and its applications; 2014, Springer. p. 1225—1234.

[30] "Comparison of genetic algorithm and simulated annealing technique for optimal path selection in network routing" by Nair T, Sooda K. arXiv preprint arXiv:1001, 3920; 2010.

[31] "Comparison of metaheuristic algorithms for solving machining optimization problems" by Madic M, Markovic D, Radovanovic M. FactaUniv Series: MechEng 2013;11(1):29–44.

[32] "A bi-objective workflow application scheduling in cloud computing systems" by Aryan Y, Delavar AG.

[33] "A learning architecture for scheduling workflow applications in the cloud" by Barrett E, Howley E, Duggan J. In: Proceedings of the 2011 ninth IEEE European conference on web services (ECOWS); 2011. IEEE.

[34] "Budget constrained, priority based genetic algorithm for workflow scheduling in cloud" by Verma A, Kaushal S. In Proceedings of the Fifth International Conference on Advances in Recent Technologies in

Communication and Computing (ARTCom 2013); 2013.

[35] "Ant colony optimization" by Yaseen SG, Al-Slamy NM. IJCSNS 2008;8(6):351.

[36] "Deadline and cost based ant colony optimization algorithm for scheduling workflow applications in hybrid cloud" by Singh L, Singh S.

[37] "Score based deadline constrained workflow scheduling algorithm for Cloud systems" by Singh R, Singh S. Int J Cloud Comput: Sen' Archit 2013;3(6):31—41.

[38] "Scheduling workflow in cloud computing based on ant colony optimization algorithm" by Zhou Y, Huang X. In: 2013 sixth international conference on proceedings of the business intelligence and financial engineering (BIFE); 2013. IEEE.

[39] "A multiple pheromone algorithm for cloud scheduling with various QQS requirements" by Gogulan R, Kavitha A, Karthick Kumar U. IntJ ComputSci 2012;9:3.

[40] "A load balancing model using firefly algorithm in cloud computing" by A.Paulin Florence, V.Shanthi Journal of computer science 2014.