ISSN No. 0976-5697

Volume 8, No. 3, March – April 2017

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

Comparative Analysis of Job Scheduling Algorithms in Parallel and Distributed Computing Environments

Navjot Kaur
Computer Engineering and Technology Department
Guru Nanak Dev University
Amritsar, India

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

Abstract: Due to an unprecedented increase in the number of computing resources in different organizations, effective job scheduling algorithms are required for efficient resource utilization. Job scheduling in considered as NP hard problem in parallel and distributed computing environments such as cluster, grid and clouds. Metaheuristics such as Genetic Algorithms, Ant Colony Optimization, Artificial Bee Colony, Cuckoo Search, Firefly Algorithm, Bat Algorithm etc. are used by researchers to get near optimal solutions to job scheduling problems. These metaheuristic algorithms are used to schedule different types of jobs such as BSP, Workflow and DAG, Independent tasks and Bag-of-Tasks. This paper is an attempt to provide comprehensive review of popular nature-inspired metaheuristic techniques which are used to schedule different categories of jobs to achieve certain performance objectives.

Keywords: scheduling, metaheuristics, multi-criteria, metrics, BSP, workflow, independent tasks, bag-of-tasks

I. INTRODUCTION

Various researchers have shown a keen interest in scheduling strategies due to exponential increase in number of resources in different organizations. Scheduling allows optimal resource allocation among various tasks in a finite time resulting in quality service [1]. Scheduling and resource allocation are considered as NP hard problems. Population based metaheuristics are found to be effective for finding optimal or near optimal solutions. In this paper, various types metaheuristic techniques are discussed and various objective functions that researchers have worked on, using these techniques in different types of environments are discussed.

A. *Scheduling*

The method by which specified work is assigned to the resource by some means, to complete a particular task is called scheduling. Schedulers keep the resources busy by allowing multiple users to share the systems resources in an effective manner. It is due to scheduling that computers are able to multi-task within a single CPU. Distributed computing has grabbed much attention in the recent years as it is reliable, highly scalable and has low cost [1]. The complexity of scheduling increases due to heterogeneity of the processing and communication resources, which leads to NP-Hard problems. For such problems, there aren't any algorithms that may produce optimal solutions. This brings us to metaheuristic algorithms that offer near optimal solutions within reasonable time. There are various types of metaheuristic algorithms namely Ant Colony Optimization (ACO), Firefly Algorithm, Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Cuckoo Search, Simulated Annealing etc.

B. *Metaheuristics*

A metaheuristic is a procedure that helps to find or generate an algorithm that may help to generate optimal solutions. Metaheuristics are of various types based on search strategy, single solution based, nature inspired etc. Further are discussed various metaheuristic algorithms:

1) Genetic Algorithm: It is an optimization method based on population and is based on Darwin's theory of evolution [2]. In GA, each possible solution is represented by a chromosome. An initial population is taken randomly and it is used as a starting point. A fitness function is calculated for each chromosome so that it is known whether the chromosome is suitable or not. Crossover and mutation functions are performed on the selected chromosmes and offsprings for new population are created. This process is repeated until enough offsprings are created [3]. The basic steps for genetic algorithm are :

- Intialisation
- Selection
- Crossover
- Mutation

1. Pick an initial arbitrary population of individuals
2. Evaluate the fitness function of the individuals
3. **repeat**
4. The best individuals should be selected
5. Generate new individuals by applying crossover and mutation operators
6. The fitness function for new individuals should be examined
7. The worst individuals are replaced with the best ones
8. **until** a stopping criteria is met

Figure 1. Pseudo code for GA

2) Ant Colony Optimization: In ACO, a number of artificial ants help in building solution for optimization problems and via a communication scheme, they exchange the information. They find the shortest paths as the moving ants lay pheromone on the ground, so that when another ant encounters it, it can detect it and decide to follow the trail. As a result, the emerged collective behavior is an indication that if a number of ants choose a particular path, then the probability of other ants following the same path increases [4]. The main idea of ACO is to model a problem as the search of minimum cost path in a graph [5].

© 2015-19, IJARCS All Rights Reserved

```
1. Begin
2. Initialise pheromone trails;
3. Produce an underlying population of a solutions(ants);
4. For every ant f ϵ a: calculate fitness function (f);
5. Determine the best position for each ant;
6. Determine the best global ant;
7. Update the pheromone ;
8. Determine if termination= true;
```

Figure 2. Pseudo code for ACO

3) Artificial Bee Colony Algorithm: This is a popular approach for optimization that simulates the intelligent foraging nature of honeybees. İn this algorithm, there are three types of bees. The first ones being the employed bees. They search for food around the food source and also they share this information about the food source with the onlooker bees. They filter out the good food sources amongst those found by the employed bees. The high quality (fitness) food source is more likely to be selected. The employed bees which abandon the food source and search new ones are called scout bees [6].

```
1. Initialise a random population
2. Evaluate its fitness function
3. While (stopping criterion is not met)
4.    Pick sites for neighbourhood search
5.    The bees for picked sites should be examined and fitness function is calculated
6.    From each patch, the fittest bee should be selected.
7.    Remaining bees are assigned to search randomly and evaluate their fitness
8. End while
```

Figure 3. Pseudo code for ABC

4) Firefly Algorithm: It is inspired from the flashing behavior of tropical fireflies. It has the following important features:

- The fireflies are attracted to each other irrespective of their gender.
- A firefly's attractiveness increases with its flash's brightness and attractiveness and brightness decrease with increase in distance.
- The illumination or brightness of a firefly is afflicted by the landscape of objective function[7].

```
1. Define objective function f(a), where a = (a1,..... ad)
2. Produce an underlying population of fireflies
3. Formulate the light intensity L
4. Specify the absorption coefficient β
5. While (t< Max_Gen)
6. For i=1 to n (all n fireflies)
7.    For j=1 to n (all n fireflies)
8.       If ( Lj > Li ), move firefly i towards firefly j
9.    End if
10.      Examine new solutions and update light intensity;
11.   End for j
12. End for i
13. Rank the fireflies and find the current best
14. End while
```

Figure 4. Pseudo code for firefly algorithm

5) Cuckoo Search Algorithm: It is inspired by brood parasitism of some cuckooo species. They lay eggs in the nests of ther birds of different species. Below are the three basic rules for this algorithm:

- One egg is laid at a given time and dumped in a randomly selected nest.
- Only the nest having the higher quality eggs are moved to the next generation.
- The number of host nests is fixed and probability of discovering the egg laid by the cuckoo by the host bird is p_a. The host can possibly discard the egg or it can abandon the nest and build a fresh one [7].

```
1.  begin
2.   Define objective function f(o)
3.   Formulate the initial population of h host nest
4.   Eggs should be ranked after assessing the fitness
5.   while (z>MaxGeneration) or stopping criteria is met
6.    z= z+1
7.   Get a cuckoo arbitrarily or produce new solution by Levy flights
8.    Assess fitness, Fi
9.    Choose a nest j, randomly
10.    if ( Fi > Fj )
11.       Replace j with the new solution
12.   end if
13.    Abandon the worst net with probabilty Pa and a new nest is then built
14.    Assess fitness, rank the solutions and find the current best
15.   end while
16: end
```

Figure 5. Pseudo code for cuckoo search

6) Particle Swarm Optimization Algorithm: The idea of PSO emerged from the swarming behavior of flock of birds, swarm of bees, schools of fish etc. İt is applied to solve different function optimization problems. İn PSO, the solutions are named particles that travel in the problem space. They follow the present optimum particles. The co-ordinates of each particle in the problem space are tracked by the particle. They are associated with the best solution achieved up to now and the value is known as pbest. Another best value, lbest, is the best value attained by any particle in the neighbourhood of the particle. Global best value, gbest, is obtained when a particle takes all the population as its neighbours [8].

```
1.   For each particle
2.      Initialise particle
3.   End for
4.   Do
5.      For each particle
6.         Evaluate the value of fitness
7.            if the fitness value is better than the best value
8.               set current value as pbest
9.         End
10.        select the particle with best fitness value as gbest
11.     For each particle
12.       evaluate particle velocity
13.       update particle position
14.     End
15.  while maximum iterations or minimum error condition is not achieved.
```

Figure 6. Pseudo code for PSO

© 2015-19, IJARCS All Rights Reserved

7) Simulated Annealing: This technique is inspired from annealing in metallurgy. İt involves heating and then slow cooling of a material. This is used to remove/reduce the defects in a crystal. To accomplish this, the material is annealed i.e. it is heated to a specific temperature and then cooled slowly until the material freezes into the form of a good crystal [9]. The slow cooling implies to slow decrease in probablility of accepting worse solutions. İf the cooling is slow, it means that there is a higher chance of finding optimal or near-optimal solutions [10].

```
1.  S0 = GenerateSolution()
2.  Temp = START_TEMP
3.  K = 0 // iteration count
4.  while
5.      S1 = Neighbour (S0)
6.      if (Fitness (S1) < Fitness (S0))
7.          S0 = S1
8.      else if (rand() < tempFunc (S0, S1, Temp,  K))
9.          S0 = S1
10.     end if
11.     AnnealingSchedule (S0, Temp, K)
12.     K = K+1
13. end while
```

Figure 7. Pseudo code for simulated annealing

The following table consists of the comparison of different nature-inspired metaheuristic algorithms. They are compared on the basis of their merits and demerits.

Table I. A Comparison Of Various Nature-Inspired Metaheuristic Algorithms.

Sr. No.	***Technique***	***Pros***	***Cons***
1.	Genetic Algorithm	Finds near optimal solution, Avoids being trapped in local optimal solutions.	No guarantee of finding global maxima, Convergence time is more.
2.	Ant Colony Opt. Algorithm	Inherent parallelism, Can be used in dynamic application.	Time to convergence is uncertain.
3.	Artificial Bee Colony Algorithm	High Performance, Fewer control parameters, Easily modified and hybridized with other metaheuristic algorithms.	Risk of losing relevant information. Population of solution increases the computational cost.
4.	Firefly Algorithm	All advantages of swarm based algorithms, Precise and robust.	Trapped in local minima, Slow convergence.
5.	Cuckoo Search	Global convergence, Global optimality.	Doesn't incorporate any type of local search to increase the convergence speed, Performance highly dependent on α i.e. step size.
6.	Particle Swarm Opt. Algorithm	Fast search speed, Calculation is simple.	Tendency of premature convergence, Suffers from partial optimism.
7.	Simulated Annealing	Can deal with arbitrary systems and cost functions, Gives a "good" solution.	Few local minima, Slow.

C. Multi-criteria

Multi-criteria or multi-objective optimization is a type of decision making criteria. In this, two or more than two objective functions are optimized simultaneously. It comes in use when optimal decisions are required to be taken in presence of trade-offs between two or more contradictory functions. In mathematical terms, it can be written as:

$$min(g_1(x), g_2(x),\ldots.g_a(x))$$
$$s.t.\ x \epsilon X$$

where the integer $\boldsymbol{a \geq 2}$ and $\boldsymbol{a}$ is the number of objectives and $\boldsymbol{X}$ is the set of feasible decision vectors. This value of $\boldsymbol{a}$ indicates that minimum 2 objectives are required.

In this work, various types of jobs, parallel or non-parallel are studied and based on those, tables have been constructed. Main focus is on the objectives, type of job and environment. Mainly four types of jobs have been studied, namely: Bulk Synchronous Parallel (BSP), DAG & Workflow, Independent Jobs and Bag-of-Tasks.

II. PERFORMANCE METRICS

Before going further, below are some performance metrics that are desired by different types of users and providers:

A. ***Makespan:*** Makespan refers to the total length of the schedule i.e. the finishing time of the last task. İt is the most popular optimization criterion and indicates the productivity of a system. Lesser the value of makespan, more efficient is the scheduler [11].
Makespan= $Max_{k\epsilon tasks}\ (T_k)$, where T_k is the finishing time of task k.

B. ***Flowtime:*** The sum of finalisation times of all tasks is called flowtime.
Flowtime= $\sum_{k\epsilon tasks}(T_k)$, where T_k is the finishing time of task k.

C. ***Convergence Speed:*** The speed at which an iterative sequence converges to a point where it reaches an optimal or near optimal solution is called the speed of convergence.

D. ***Response Time:*** Sum of waiting time and execution time is called the response time.

E. ***Throughput:*** The number of jobs that complete their execution per unit time is termed as throughput. İn other words, it is the amount of work done by a system in given time.

F. ***Speedup:*** It is a process of increasing the performance of a system. When there is any enhancement in the resources, speedup shows the effect on the performance of a system.

G. ***Scalability:*** Generally, speedup declines when number of processors increase and size of problem is fixed. Scalabilty refers to the change in performance of a parallel system as the size of problem and computer size increase [11].

H. ***Fairness:*** Fairness means that every job should get equal share of CPU time and no job should face starvation.

I. ***Energy Consumption:*** It is the amount of power or energy used by a system. It is preferable that a system uses less energy because it is more economical.

J. ***Budget Constraint:*** It represents the total cost restriction for executing all jobs.

© 2015-19, IJARCS All Rights Reserved

III. BULK SYNCHRONOUS PARALLEL (BSP) MODEL BASED JOBS

A Bulk Synchronous Parallel (BSP) computer comprises of a collection of processors. Each processor has its own memory i.e. it is a distributed-memory computer. It has mainly three components: processor/memory pair, communication network and synchronisation barrier [12]. This model is used for parallel jobs. The jobs in BSP comprise of a fixed number of tasks which have almost identical requirements and each task constitutes various iterations. Below is the table for comparison of BSP based jobs in different environments.

TABLE II. Comparison of Various BSP Based Algorithms

Ref. No.	*Technique*	*Year*	*MO*/SO***	*Environment*	*Remarks*
[12]	MaMR	2017	MO(Speedup, Scalability)	Cloud	MaMR model supports map & reduce function; new merge phase is added to MapReduce.
[13]	MigPF (Migration Model)	2016	SO(Load Balancing)	Heterogeneous Environment	MigBSP is redesigned to propose MigPF.
[14]	Genetic Algorithm	2016	MO(Energy Consumption, Makespan, Convergence Speed)	Heterogeneous Environment	Multi-objective genetic algorithm based on weighted blacklist.
[15]	Multi-memory BSP (MBSP)	2015	MO(Cost, Communication Time, Memory Access Time)	Parallel Computing	An extension to BSP model, MBSP is proposed.
[16]	Genetic Algorithm	2015	MO(Makespan, Flowtime)	Multi-cluster	GA based scheduling metaheuristic for large scale multi-cluster environments; applying co-allocation.
[17]	BSP-based Model	2015	MO(Execution Time, Solution Accuracy)	GPU	Simple BSP model for performance prediction.
[18]	Genetic Algorithm	2014	MO(Solution Accuracy, Execution Time)	Hadoop	Improving population diversity in GA using migration technique.
[19]	Parallel Processing of Graphs	2014	MO(Computation Time, Scalability)	Cloud	Comparison of MR (MapReduce), MR2 (extension of MR) & BSP.
[20]	Distributed-memory and Shared-memory Model	2013	MO(Speedup, Scalability, Performance Prediction)	Cloud	Proposing BSPCloud- hybrid of distributed memory and shared memoryodel.
[21]	Parallel GPU Model	2012	SO(Degree of Optimality)	GPU	Parallel GPU model for GPU algorithm development.

*multi-objective **single-objective

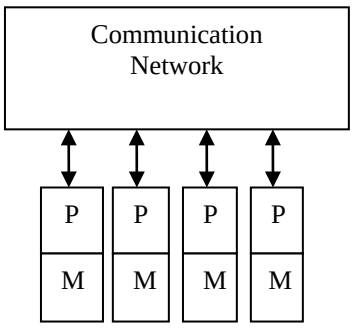

Figure 8. BSP model

Weipeng and Danyu [12] presented a new programming model named MaMR for cloud applications. It supports multiple map and reduce functions that run in parallel. It makes use of a hybrid shared-memory BSP model. The simulation results show that the presented model shows effective improvement in performance compared to the previous work. Rodrigo et al. [13] proposed a migration model, MigPF, for BSP programs in heterogeneous environments. A prototype was developed with the AMPI library and was tested against other built-in AMPI rescheduling policies. Experimental results showed 41% performance gain and the overhead limited to 5%. Gabaldon et al. [14] presented a multi-objective Genetic Algorithm that is based on a weighted blacklist. The authors have worked on reducing makespan and also towards energy conservation. An extension to the BSP model called MBSP, for multi-memory BSP, in the presence of multiple heirarchies and cores was proposed in [15]. Gabaldon [16] worked towards the scheduling in multi-cluster environments using a metaheuristic technique i.e. Genetic Algorithm. The multi-objective function optimises makespan as well as the flowtime. Marcos et al. [17] proposed a simple BSP-based model for predicting execution times for CUDA applications on GPUs . The model predictions were 0.8 to 1.2 times the measured execution times which shows that the model is good enough for generalising predictions for different problem sizes and GPU configurations. The population diversity in Genetic Algorithm is improved in [18]. The main focus is on migration. Three techniques for parallelisation were used which were : MapReduce, BSP and MPI. The proposed method was compared with existing methods and results showed improvement in solution accuracy. Three approaches for graph processing namely MapReduce, BSP

© 2015-19, IJARCS All Rights Reserved

and MR2 were compared by Tomasz [19]. The experiments were conducted on four large graph datasets. Xiaodong and Weikin [20] proposed an integration of distributed-memory and shared-memory BSP model named BSPCloud. The BSPCloud can make full utilisation of multi-core clusters. The performance predictability and speedup were evaluated. J. Steven and Ovidiu [21] presented a parallel GPU model which helps the parallel GPU algorithm designers to design and implement optimal algorithms. Results show that algorithms designed based on the model's principles give significant increase in performance.

IV. WORKFLOW AND DAG BASED JOBS

Workflow scheduling can be called as a general form of task scheduling. Here, the applications are modelled as Directed Acyclic Graphs (DAG). DAG is a popular representation of workflow applications. Workflow scheduling can broadly be classified into two categories:

A. ***Static Workflow Scheduling:*** Static scheduling can search from the solution space globally at workflow level. But its limitation is that it assumes that accurate task communication and execution time could be obtained before scheduling which can not necessarily always be true [22].

B. ***Dynamic Workflow Scheduling:*** It is helpful when scheduling information is unavailable or when there is resource contention with other system's load [23].

Cuicui et al. [24] proposed a new bacterial foraging optimization algorithm, named BFO-CC. It is a very competitive algorithm that measures basis-vector, non-uniform step-size and conjugation strategies. Simulation results show an improvement in convergence, solution quality and computational efficiency. Neetesh and Deo [25] proposed a hybrid PSO-GA metaheuristic for DAG scheduling. İts main aim is to improve the solution obtained by PSO using GA. The hybrid is tested for two linear algebra problems namely LU decomposition and Gauss-Jordan elimination. Dzmitry et al. [26] address the performance issues of resource allocation in cloud. A communication-aware model called CA-DAG is proposed for making separate resource allocation decisions. Fengyu et al. [27] present a workflow task scheduling algorithm based on fuzzy clustering of resources called FCBWTS. Comparisons show that the algorithm is better than HEFT and DLS algorithms both in makespan as well as time consumed. A new evolutionary algorithm named CSA based on Cuckoo Search is proposed in [28]. It is based on Levy flight behavior and obilgate blood behavior of some cuckoo species. The results of experimental evaluation show that when P_a value is less, speed and coverage of the algorithm become very high. Yuming et al. [29] developed an improved Chemical Reaction Optimization (CRO) called HCRO (hybrid CRO).

TABLE III. Comparison of DAG and Workflow Based Jobs

Ref. No.	*Technique*	*Year*	*MO/SO*	*Type of Parallel Job*	*Environment*	*Remarks*
[24]	Bacterial Foraging Optimization Algorithm	2016	MO(Solution Quality, Computational Efficiency)	Workflow	Parallel Computing	A new designed chemotaxis and conjugation strategy for BFO named BFO-CC.
[25]	Particle Swarm Opt. & Genetic Algorithm	2016	MO(Solution Quality, Scalability, Makespan)	DAG	Multiprocessor System	Hybrid of GA and PSO for scheduling DAG.
[26]	Communication Aware Modeling	2016	MO(Approximation Factor, Efficiency of Produced Schedule)	DAG	Cloud	CA-DAG allows making separate resource allocation decisions.
[27]	Workflow Task Scheduling	2015	MO(Makespan, Speedup)	DAG	Cloud	FCBWTS algorithm to minimise makespan and comparing it with HEFT and DLS algorithms.
[28]	Cuckoo Search	2015	MO(Fitness Fucntion, Flowtime, Speed &Coverage of algorithm)	Workflow	Cloud	Simple cuckoo search.
[29]	Chemical Reaction Optimisation	2014	MO(Makespan, Speed of Convergence)	DAG	Heterogeneous Computing Systems	Hybrid chemical reaction optimisation (HCRO) is proposed along with a new selection strategy.
[30]	Firefly Algorithm	2014	MO(CPU Utility Rate, Memory Usage Rate, Load Balancing)	Workflow	Cloud	Load balancing by dealing with a set of requests & servers; servers are hence assocciated with nodes and each node has some attributes.
[31]	Particle Swarm Opt & Genetic Algorithm	2013	MO(Makespan, Flowtime)	DAG	Heterogeneous Distributed Systems	PSO–based GA; solutions initialised using some effective list scheduling strategy, evolved using crossover and mutation operator; PSO used for guiding the search effectively.
[32]	Ant Colony Optimization, Cuckoo Search	2012	SO(Makespan)	Workflow	Parallel Computing	Hybrid proposed by combining the merits of ACO and Cuckoo Search.
[33]	Energy Aware Genetic Algorithm	2011	MO(Makespan, Energy Consumption)	DAG	Cloud	Bi-objective GA based on Dynamic Voltage Scaling (DVS).

© 2015-19, IJARCS All Rights Reserved

Simulations show that the presented algorithm is better than existing algorithms at scheduling DAG tasks and gives lesser makespan and better speed of convergence. A. Paulin [30] focussed on load balancing using firefly algorithm in cloud computing environments. The simulation gave expected results thus proving the proposed approach to be PSO-based hybrid with GA for task scheduling in DAGs is presented in [31]. Babukartik and Dhavachelvan [32] proposed a hybrid by combining the merits of ACO and Cuckoo search. The simulation results show that with the increase in number of tasks, task creation time and results retrieval is also increased. Mezmaz et al. [33] explored the problem of scheduling precedence-constrained parallel applications. A parallel bi-objective genetic algorithm was proposed that considers both makespan and energy consumption. A comparison of DAG and workflow based jobs is shown in Table III.

V. INDEPENDENT JOBS

Independent jobs are those in which the tasks do not directly communicate with each other. All tasks perform same or similar function and it is not necessary for them to run simultaneously.

Medhat et al. [34] compared a cloud task scheduling algorithm based on Ant Colony Optimization (ACO) with FCFS and RR scheduling algorithm. The main motive is to minimize the makespan. The simulation in CloudSim showed that the ACO based cloud task scheduling performs better than FCFS and RR algorithms. Rajni et al. [35] proposed a PSO-based hyper-heuristic resource scheduling algorithm for job scheduling in Grid environments. The PSOHH gave better results than existing hyper-heuristic scheduling algorithms as far as cost, time and makespan are concerned. A hybrid PSO and BAT algorithm is presented in [36]. The hybrid combines the features of both PSO and BAT algorithm. Results give faster convergence speed, less parameters to tune and easy searching in large spaces. An improved version of Ant Colony Optimization algorithm is presented in [37]. This work considers SLA violation, energy consumption and load balancing. Ch. Srinivasa [38] proposed a fuzzy differential evolution for scheduling on computational grids. The performance of fuzzy DE was compared with other existing evolutionary algorithms. Experimental results show that the proposed algorithm produced more optimal solutions compared to other algorithms. Rajni [39] proposed a novel bacterial foraging based hyper heuristic scheduling algorithm for scheduling resources in grid environment. The simulation results give better makespan and minimized cost. Susmita et al. [40] present a resource broker architecture for GA in computational grids. There are multiple users that submit the jobs and multiple providers that are selected by the broker. As a result, the total completion time (TCT) is minimized. Jinn-Tsong et al. [41] proposed an improved differential evolution algorithm (IDEA) for task scheduling and resource allocation in cloud computing environment. The proposed algorithm integrates the Taguchi method and Differential Evolution Algorithm. Results give smaller makespan and cost. Dasgupta et al. [42] focussed on load balancing using genetic algorithm while scheduling tasks in cloud.

TABLE IV. Comparison of Algorithms Based on Independent Jobs

Ref No.	*Technique*	*Year*	*MO/SO*	*Environment*	*Remarks*
[34]	Ant Colony Opt.	2015	SO(Makespan)	Cloud	Task scheduling based on ACO is compared with FCFS and RR with main goal being makespan minimisation.
[35]	Particle Swarm Opt.	2015	MO(Makespan, Cost Reduction of Job Execution)	Grid	PSO based hyper heuristic is compared with existing common heuristic scheduling algorithms.
[36]	Particle Swarm Opt. and BAT Algorithm	2015	MO(Convergence Speed, Energy Conservation)	Cloud	Hybrid of PSO-MOBA; M/M/m queuing model for managing queues of job and average execution time.
[37]	Ant Colony Opt.	2014	MO(Load Balancing, SLA Violation, Energy Consumption)	Cloud	SALB for load balancing by finding overloaded nodes in minimum time.
[38]	Fuzzy Differential Evolution	2014	SO(Makespan)	Grid	Fuzzy based DE algorithm compared with GA, SA, DE and fuzzy PSO.
[39]	Bacterial Foraging Optimization Algorithm	2013	MO(Cost of the executing job, Makespan)	Grid	Hyper heuristic BFO; performance is analysed by varying both number of jobs and resources.
[40]	Genetic Algorithm	2013	SO(Total Completion Time of all jobs)	Grid	GA based resource broker with multiple users and resource providers.
[41]	Differential Evolution	2013	MO(Resource Renting Cost, Makespan)	Cloud	Taguchi Method and DE combined to propose Improved Differential Evolution Algorithm (IDEA) for finding potential offsprings.
[42]	Genetic Algorithm	2013	MO(Load Balancing, Makespan)	Cloud	Basic GA.
[43]	Artificial Bee Colony Algorithm	2013	MO(Makespan, Convergence Speed)	Grid	Binary ABC, BABC, is proposed for binary integer job scheduling problems; further an extension of BABC is also proposed.
[44]	Genetic Algorithm and Variable Neighbouhood Search	2012	MO(Execution Cost, Makespan)	Grid	GA is main algorithm, VNS is for improving individuals in population.

© 2015-19, IJARCS All Rights Reserved

The algorithm is compared with FCFS, RR and SHC and results show that the proposed algorithm shows a decrease in response time. Kim et al. [43] proposed a Binary Artificial Bee Colony (BABC) algorithm for binary integer job scheduling in grid environment. Further an extension to BABC is proposed that makes use of a flexible ranking strategy (FRS) to generate and use solutions. Sara et al. [44] proposed a hybrid genetic algorithm and variable neighbourhood search as a hope of reducing the oevrall execution cost without much increase in makespan. Experiments show that the hybrid performs well in case of execution cost. In the worst case, increase in makespan was less than 17% which is tolerable.

VI. BAG-OF-TASKS

Bag-of-Tasks constitute of many independent tasks that can be excuted in parallel. BoT find their use in many field like image processing data mining etc. In these fields, applications consist of several steps instead of a single step. These steps could be sequential, parallel or hybrid of both. Each step processes a BoT [45]. The tasks of BoT applications are independent of each other and may have different computational requirements. They also do not need to communicate with each other during execution [46]. BoT execution usually requires costly investments in infrastructure, the reason being high consumption of resources and parallel nature of BoTs [49].

Zhicheng et al. [45] focus on fulfilling the workflow deadline by using a dynamic cloud resource provisioning and scheduling algorithm. The VMs are rented dynamically and main motive is to minimize the resource renting cost. George and Helen [46] proposed power-aware scheduling policies by extending the Min-Min and Max-Min policies for homogeneous clusters. They proposed two types of policies namely greedy and conservative. The greedy policies achieved upto 20.6% energy consumption which was more than conservative policies. The execution of BoTs and high-priority tasks wasn't greatly affected. Ioannis [47] worked towards scheduling of bag-of-tasks applications in heterogeneous cloud with the use of metaheuristic techniques. Two algorithms, Simulated Annealing and Tabu search have been applied . Simulation results for both the algorithms show that there were benefits in both cost and performance. Javier [48] proposed a decentralised model for scheduling policy whose sole objective is fairness. Simulation results prove this policy to be scalable and effective. The proposed policy performed only 11% worse than centralised implementations. A new cloud BoT

TABLE V. Comparison of Bag-of-Tasks Based Algorithms

Ref No.	Technique	Year	MO/SO	Environment	Remarks
[45]	Delay-based Dynamic Scheduling	2017	MO(Resource Renting Cost, Meeting Workflow Deadline)	Cloud	Using ElasticSim and VM's are rented dynamically.
[46]	Power Aware Scheduling	2016	MO(Energy Conservation, Minimising Finishing Time)	Heterogeneous Cluster	Min-Min and Max-Min scheduling policies are extended & proposing power aware centralised scheduling policies.
[47]	Tabu Search & Simulated Annealing	2015	MO(Makespan, Utilisation,Total Trace Cost, Cost Performance Efficiency)	Cloud	Multiple job arrival levels are supported.
[48]	Fair Share Policy	2015	MO(Scalability, Maximum Stretch)	Large-scale Platform	Tries to provide same share to each application; amount of computation required by each user is considered.
[49]	Multiagent Systems	2015	MO(Chargeable Allocation Time, Concurrent executions)	Cloud	Consumers can reduce costs by adopting CNP; using CloudAgents can execute BoTs in concurrent manner.
[50]	Automatic Tuning	2015	MO(Overhead, Speedup)	Aplug	The automatic tuning framework adapts degree of parallelism; enables users to utilise CPU resources efficiently.
[51]	Simulated Annealing	2015	MO(Makespan, Normalised Schedule Length , Total Trace Cost)	Cloud	Two SA algorithms' performance is evaluated taking virtual machines' heterogeneity into account.
[52]	Energy Efficient Scheduling	2014	SO(Energy Conservation)	Cloud	Intelligent scheduling combined with Dynamic Voltage & Frequency Scaling (DVFS).
[53]	Genetic Algorithm	2012	MO(Budget Constraint, Deadline Constraint)	Cloud	GA for selecting sub-optimal sets of resources.
[54]	Rescheduling	2012	MO(Response Time, Slowdown Reduction)	Multi-provider	Handling inaccurate run-time estimates.
[55]	Scheduling and Task Partitioning	2011	SO(Scalability)	Heterogeneous Platforms	Multi-node systems.

© 2015-19, IJARCS All Rights Reserved

execution tool, CloudAgent is proposed in [49] which can handle concurrent BoT executions and bear low execution costs. Majed [50] proposed a framework for automatic tuning named APlug that collects sample execution times and builds a model. Experiments were run on 16,384 CPUs, 480 cores on Linux cluster and 80 cores on Amazon EC2. The results showed that APlug is very accurate with minimal overhead. Ioannis [51] evaluated the use of simulated annealing and thermodynamic simulated annealing for scheduling in multi-cloud system along with virtual machines. The heuristics consider multiple objectives while scheduling and try to optimize both cost and performance. Calheiros and Buyya [52] target the issue of enery-efficient execution of urgent , CPU-intensive Bag-of-Tasks applications. A cloud-aware scheduling algorithm was proposed that applies DVFS for enabling deadlines and thus reducing energy consumption. J. Octavio and Kwang [53] proposed a genetic algorithm for both budget and deadline constrained execution of BoT applications. Marco [54] in this work handle inaccurate run-time estimates by proposing a coordinated rescheduling algorithm. Experiments were performed using Grid'5000. The results showed 5% reduction in response time and 10% for slowdown metrics. Silva [55] studied the scalability of BoT applications running on multi-node systems. Table V. includes the comparison of various Bag-of-Tasks based algorithms.

VII. CONCLUSION

This paper gives a wide review of different types of metaheuristic techniques which are used by researcher to obtain near optimal solutions of four different types of jobs namely BSP, Workflow and DAG, Independent tasks and Bag-of-Tasks. The various issues and problems found in the individual metaheuristics can be overcome by using the hybridization of two or more metaheuristic techniques. In this paper, an attempt has been made to provide the review and comparison of different metaheuristic algorithms which are used to schedule individual type of jobs.

VIII. REFERENCES

[1] Kalra, Mala, and Sarbjeet Singh. "A review of metaheuristic scheduling techniques in cloud computing." *Egyptian informatics journal* 16.3 (2015): 275-295.

[2] Roberge, Vincent, Mohammed Tarbouchi, and Gilles Labonté. "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning." *IEEE Transactions on Industrial Informatics* 9.1 (2013): 132-141.

[3] Garg, Richa, and Saurabh Mittal. "Optimization by genetic algorithm." *International Journal of Advanced Research in Computer Science and Software Engineering* 4.4 (2014): 587-589.

[4] Yaseen, Saad Ghaleb, and Nada MA Al-Slamy. "Ant colony optimization." *IJCSNS* 8.6 (2008): 351.

[5] Selvi, V., and Dr R. Umarani. "Comparative analysis of ant colony and particle swarm optimization techniques." *International Journal of Computer Applications (0975–8887)* 5.4 (2010).

[6] Yunfeng Xu, Ping Fan, and Ling Yuan, "A Simple and Efficient Artificial Bee Colony Algorithm," Mathematical Problems in Engineering, vol. 2013, Article ID 526315, 9 pages, 2013. doi:10.1155/2013/526315

[7] Kaur, Navneet, and Amit Chhabra. "Analytical review of three latest nature inspired algorithms for scheduling in clouds." *Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on*. IEEE, 2016.

[8] Aote, Shailendra S., M. M. Raghuwanshi, and Latesh Malik. "A brief review on particle swarm optimization: limitations & future directions." International Journal of Computer Science Engineering (IJCSE) 14.1 (2013): 196-200..

[9] Rutenbar, Rob A. "Simulated annealing algorithms: An overview." *IEEE Circuits and Devices Magazine* 5.1 (1989): 19-26.

[10] Xinchao, Zhao. "Simulated annealing algorithm with adaptive neighborhood." *Applied Soft Computing* 11.2 (2011): 1827-1836.

[11] Xhafa, Fatos, and Ajith Abraham. "Computational models and heuristic methods for Grid scheduling problems." *Future generation computer systems* 26.4 (2010): 608-621.

[12] Jing, Weipeng, et al. "MaMR: High-performance MapReduce programming model for material cloud applications." *Computer Physics Communications* 211 (2017): 79-87.

[13] da Rosa Righi, Rodrigo, et al. "MigPF: Towards on self-organizing process rescheduling of Bulk-Synchronous Parallel applications." *Future Generation Computer Systems* (2016).

[14] Gabaldon, Eloi, et al. "Blacklist muti-objective genetic algorithm for energy saving in heterogeneous environments." *The Journal of Supercomputing* (2016): 1-16.

[15] Gerbessiotis, Alexandros V. "Extending the BSP model for multi-core and out-of-core computing: MBSP." *Parallel Computing* 41 (2015): 90-102.

[16] Gabaldon, E., et al. "Multi-criteria genetic algorithm applied to scheduling in multi-cluster environments." *Journal of Simulation* 9.4 (2015): 287-295.

[17] Amaris, Marcos, et al. "A simple bsp-based model to predict execution time in gpu applications." *High Performance Computing (HiPC), 2015 IEEE 22nd International Conference on*. IEEE, 2015.

[18] Enomoto, Takuto, and Masaomi Kimura. "Improving Population Diversity in Parallelization of a Real-Coded Genetic Algorithm Using MapReduce."

[19] Kajdanowicz, Tomasz, Przemyslaw Kazienko, and Wojciech Indyk. "Parallel processing of large graphs." *Future Generation Computer Systems* 32 (2014): 324-337.

[20] Liu, Xiaodong, et al. "BSPCloud: A hybrid distributed-memory and shared-memory programming model." *International Journal of Grid and Distributed Computing* 6.1 (2013): 87-97.

[21] Kirtzic, J. Steven, Ovidiu Daescu, and T. X. Richardson. "A parallel algorithm development model for the GPU architecture." *Proc. of Int'l Conf. on Parallel and Distributed Processing Techniques and Applications*. 2012.

[22] Wu, Fuhui, Qingbo Wu, and Yusong Tan. "Workflow scheduling in cloud: a survey." *The Journal of Supercomputing* 71.9 (2015): 3373-3418.

[23] Sonmez, Ozan, et al. "Performance analysis of dynamic workflow scheduling in multicluster grids." *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010.

[24] Yang, Cuicui, et al. "Bacterial foraging optimization using novel chemotaxis and conjugation strategies." *Information Sciences* 363 (2016): 72-95.

[25] Kumar, Neetesh, and Deo Prakash Vidyarthi. "A novel hybrid PSO–GA meta-heuristic for scheduling of DAG with communication on multiprocessor systems." *Engineering with Computers* 32.1 (2016): 35-47.

[26] Kliazovich, Dzmitry, et al. "CA-DAG: Modeling communication-aware applications for scheduling in cloud computing." *Journal of Grid Computing* 14.1 (2016): 23-39.

[27] Guo, Fengyu, et al. "A workflow task scheduling algorithm based on the resources' fuzzy clustering in cloud computing environment." *International Journal of Communication Systems* 28.6 (2015): 1053-1067.

[28] Navimipour, Nima Jafari, and Farnaz Sharifi Milani. "Task scheduling in the cloud computing based on the cuckoo search algorithm." *International Journal of Modeling and Optimization* 5.1 (2015): 44.

© 2015-19, IJARCS All Rights Reserved

[29] Xu, Yuming, et al. "A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems." *IEEE Transactions on parallel and distributed systems* 26.12 (2015): 3208-3222.

[30] COMPUTING, FIREFLY ALGORITHM IN CLOUD. "A load balancing model using firefly algorithm in cloud computing." *Journal of Computer Science* 10.7 (2014): 1156-1165.

[31] Kang, Yan, He Lu, and Jing He. "A PSO-based Genetic Algorithm for Scheduling of Tasks in a Heterogeneous Distributed System." *JSW* 8.6 (2013): 1443-1450.

[32] Babukartik, R. G., and P. Dhavachelvan. "Hybrid Algorithm using the advantage of ACO and Cuckoo Search for Job Scheduling." *International Journal of Information Technology Convergence and Services* 2.4 (2012): 25.

[33] Mezmaz, Mohand, et al. "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems." *Journal of Parallel and Distributed Computing* 71.11 (2011): 1497-1508.

[34] Tawfeek, Medhat A., et al. "Cloud task scheduling based on ant colony optimization." *Int. Arab J. Inf. Technol.* 12.2 (2015): 129-137.

[35] Aron, Rajni, Inderveer Chana, and Ajith Abraham. "A hyper-heuristic approach for resource provisioning-based scheduling in grid environment." *The Journal of Supercomputing* 71.4 (2015): 1427-1450.

[36] George, Salu. "Hybrid PSO-MOBA for Profit Maximization in Cloud Computing." *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS* 6.2 (2015): 159-163.

[37] Khan, Shagufta, and Niresh Sharma. "Effective scheduling algorithm for load balancing (SALB) using ant colony optimization in cloud computing." *International Journal of Advanced Research in Computer Science and Software Engineering* 4 (2014).

[38] Rao, Ch, and B. Raveendra Babu. "A Fuzzy Differential Evolution Algorithm for Job Scheduling on Computational Grids." *arXiv preprint arXiv:1407.6317* (2014).

[39] Chana, Inderveer. "Bacterial foraging based hyper-heuristic for resource scheduling in grid computing." *Future Generation Computer Systems* 29.3 (2013): 751-762.

[40] Singh, Susmita, et al. "Genetic algorithm based resource broker for computational Grid." *Procedia Technology* 10 (2013): 572-580.

[41] Tsai, Jinn-Tsong, Jia-Cen Fang, and Jyh-Horng Chou. "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm." *Computers & Operations Research* 40.12 (2013): 3045-3055.

[42] Dasgupta, Kousik, et al. "A genetic algorithm (ga) based load balancing strategy for cloud computing." *Procedia Technology* 10 (2013): 340-347.

[43] Kim, Sung-Soo, et al. "Optimal job scheduling in grid computing using efficient binary artificial bee colony optimization." *soft computing* 17.5 (2013): 867-882.

[44] Kardani-Moghaddam, Sara, et al. "A hybrid genetic algorithm and variable neighborhood search for task scheduling problem in grid environment." *Procedia Engineering* 29 (2012): 3808-3814.

[45] Cai, Zhicheng, et al. "A delay-based dynamic scheduling algorithm for bag-of-task workflows with stochastic task execution times in clouds." *Future Generation Computer Systems* (2017).

[46] Terzopoulos, George, and Helen D. Karatza. "Power-aware Bag-of-Tasks scheduling on heterogeneous platforms." *Cluster Computing* 19.2 (2016): 615-631.

[47] Moschakis, Ioannis A., and Helen D. Karatza. "A meta-heuristic optimization approach to the scheduling of Bag-of-Tasks applications on heterogeneous Clouds with multi-level arrivals and critical jobs." *Simulation Modelling Practice and Theory* 57 (2015): 1-25.

[48] Celaya, Javier, and Unai Arronategui. "Fair scheduling of bag-of-tasks applications on large-scale platforms." *Future Generation Computer Systems* 49 (2015): 28-44.

[49] Gutierrez-Garcia, J. Octavio, and Kwang Mong Sim. "Agent-based Cloud bag-of-tasks execution." *Journal of Systems and Software* 104 (2015): 17-31.

[50] Sahli, Majed, et al. "Automatic tuning of bag-of-tasks applications." *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE, 2015.

[51] Moschakis, Ioannis A., and Helen D. Karatza. "Multi-criteria scheduling of Bag-of-Tasks applications on heterogeneous interlinked clouds with simulated annealing." *Journal of Systems and Software* 101 (2015): 1-14.

[52] Calheiros, Rodrigo N., and Rajkumar Buyya. "Energy-efficient scheduling of urgent bag-of-tasks applications in clouds through DVFS." *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*. IEEE, 2014.

[53] Gutierrez-Garcia, J. Octavio, and Kwang Mong Sim. "GA-based cloud resource estimation for agent-based execution of bag-of-tasks applications." *Information Systems Frontiers* 14.4 (2012): 925-951.

[54] Netto, Marco AS, and Rajkumar Buyya. "Coordinated rescheduling of Bag-of-Tasks for executions on multiple resource providers." *Concurrency and Computation: Practice and Experience* 24.12 (2012): 1362-1376.

[55] da Silva, Fabricio AB, and Hermes Senger. "Scalability limits of Bag-of-Tasks applications running on hierarchical platforms." *Journal of Parallel and Distributed Computing* 71.6 (2011): 788-801.

© 2015-19, IJARCS All Rights Reserved