iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 7 **Issue:** II **Month of publication:** February

DOI: http://doi.org/10.22214/ijraset.2019.2168

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 7 Issue II, Feb 2019- Available at www.ijraset.com

An Efficient Load Balancing Technique for Mutation based Particle Swarm Optimization

Neha Sethi
IKG Punjab Technical University, Jalandhar, Punjab, India.

Abstract: ***Load balancing techniques play significant role in cloud computing environment because it directly af-fects the performance of cloud data centers. An efficient load balancing technique not only provides high availability of resources to cloud users, but also enhances the performance of cloud data centers. Load balancing techniques are a typical NP-hard problem. Currently, many researchers have solved load bal-ancing problem by considering well-known metaheuristic techniques. However, these techniques suffer from one of these issues: premature convergence, poor convergence speed, initially selected random so-lutions and stuck in local optima. To handle the issues associated with existing metaheuristic techniques, in this paper, a mutation based particle swarm optimization based load balancing technique is proposed. The proposed technique has an ability to overcome several issues associated with existing techniques such as premature convergence, poor convergence speed, initially selected random solutions and stuck in local optima issues. Also, multi-objective fitness function is designed as a minimization problem. Multi-objective fitness function considers energy consumption, makespan and load imbalance rate parameters. The proposed technique outperforms existing load balancing techniques in terms of makespan, speedup, communication overheads, efficiency, utilization, mean gain time, load imbalance rate and energy consumption.***

Index terms: Load balancing, Particle swarm optimization, Cloud computing, Energy efficiency.

I. INTRODUCTION

Cloud computing is growing phase having a world view of substantial scale distributed computing in internet era [1]. It is a combination of grid and 'X' as a resources in wireless environment. Here, 'X' may be infrastructure, platform, software, data, hardware etc. [2]. The primary objective of scheduling is the assignment of jobs to available set of servers so that execution time can be minimized. Another feature of scheduling is decision-making process and is mostly used in service and manufacturing organizations. Scheduling takes place by taking help of load balancing algorithms [3]. Job is a term related with scheduling used at application level and is a script or program to execute a specific set of jobs. Load balancing algorithm is a technique to which jobs are optimistically assigned to data center resources. There is no completely perfect scheduling mechanism available due to different scheduling objectives. Scheduling algorithms can be executed or implemented under suitable conditions according to assigned applications by a good scheduler. Scheduling algorithm is a mechanism that solves a problem in seconds, minutes or even hours. Time used for execution of particular algorithm measures the efficiency of that algorithm and so time complexity can be measured from the efficiency. Time complexity plays a significant role in time execution of an algorithm. There are some time complexity algorithms used for job execution. The problem is feasible, traceable, fast and efficient in case of a program which is related to polynomial time algorithm. Using computational complexity, hypothesis number of problems can be solved as complexity class based on some resources [4].

Cloud data centers are often provisioned to handle peak loads, that can result in low resource utilization and wastage of energy. Resource utilization directly relates to energy consumption, so it should be optimized in order to save energy [5]. Cloud computing potentiality for convincing energy savings has so far been aimed at hardware features. Contrarily, software systems can also be improved at development time by postulating the energy characteristics [5].

In cloud computing, load balancing techniques are required for assigning the workload between cloud data centers to prevent a state of some nodes being over laden while others being lightly laden or even idle. The workloads should be mapped to various resources under the constraint of energy optimization [6]. With efficient load balancing, resource utilization can be enhanced which can further reduce energy consumption thereby reducing carbon emissions and cooling requirements of cloud data centers [6]. Every operational physical node in a cloud data center, produces heat. When a specific node is unreasonably used, hotspots can appear in a given data center. Therefore, not only the aspect of efficient job allocation to cloud data centers has to be considered, but also the heat generated by servers have to be measured and accounted for to avoid these hotspots [7].

As a result, there is a requirement for the development of an efficient load balancing technique that is capable of improving the performance of cloud data centers. The load balancing technique should be able to assign the workload to cloud data centers in such a way that makespan should be minimum. It should lead to the reduction in the energy consumption, thereby dropping the carbon emissions and cooling requirements of the cloud data centers, to an extent which can help to achieve green computing

[7]. Due to the factors outlined above, the energy-aware load balancing technique for cloud computing environment has been the motivation behind this work. Recently, several researchers have solved load balancing issue by considering well-known metaheuris-tic techniques. These techniques are Genetic algorithm (GA), Particle swarm optimization (PSO), Ant colony optimization (ACO), BAT algorithm, Artificial bee colony etc. However, these metaheuristic tech-niques suffer from one of these issues: premature convergence, poor convergence speed, initially selected random particles and stuck in local optima issues. Additionally, majority of existing researchers have con-sidered scheduling of independent jobs only. Also, the researchers who have focused on the dependent jobs have neglected the effect of communication overheads. Therefore, for dependent jobs it becomes more significant to minimize communication overheads. To handle these issues mutation based particle swarm optimization load balancing technique is proposed in this paper.

II. RELATED WORK

Javanmardi et al. (2014) utilized genetic based fuzzy logic to balance the jobs between available high-end servers [8]. Agarwal and Jain (2014) presented a generalized priority technique to minimize the makespan in more efficient way [9]. Patel and Bhoi (2014) enhanced priority based scheduling technique using multiple measures and attributed it to decision making model [10]. Hassan et al. (2015) utilized genetic algorithm based scheduling technique to minimize the makespan [11]. However, it suffered from poor convergence speed and is stuck in local optima issues. Kalra and Singh (2015) studied various load balancing techniques for cloud data centers using some well-known metaheuristic techniques [12]. Cho et al. (2015) proposed an integrated Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) based scheduling technique. It has ability to overcome the problem of poor convergence speed with ACO and premature convergence issue with PSO [13].

Milani and Navimipour (2016) studied various benefits and limitations of existing load balancing techniques [14]. Mittal and Katal (2016) designed optimized load balancing technique to minimize the load imbalancing between available cloud data centers [15]. Singh and Chana (2016) proved that the efficient selection of metaheuristic techniques have the ability to overcome the premature and stuck in local optima issue [16]. Masdari et al. (2017) proved that PSO based load balancing technique has good convergence speed [17]. However, it suffers from the problem of initial number of particle issue. Duan et al. (2017) proposed a machine learning technique based scheduling technique [18]. However, it was effective for those whose prior burst time is known.

Lalwani et al. (2013) designed multi-objective PSO to optimistically schedule the jobs between cloud data centers [19]. A hybrid PSO scheduling technique has been designed by considering neighborhood search strategies to achieve a trade-off among exploration and exploitation characteristics [20, 21]. Wu et al. (2014) also elaborated PSO very well. A guided PSO has been designed using mutation operator and different local search approaches to schedule the jobs between cloud data centers [22]. Behnamian (2014) designed a discrete PSO which comprises two components: a PSO and GA. The fuzzification was used to rank the jobs [23]. Zuo et al. (2014) proposed four updating techniques to dynamically update the velocity of each particle to ensure its diversity and robustness [24].

Liang et al. (2015) designed adaptive PSO based clustering to balance local and global search in an optimization process. Initially, K-means clustering divided the swarm dynamically in the whole process to construct variable subpopulation clusters and after that adopted a ring neighborhood topology for data sharing between these clusters. Then, a novel adaption technique was adopted to adjust the inertia weight of all individuals based on the states of clusters and swarms [25, 26]. Marini and Walczak (2015)

2 studied the potentials of PSO especially for scheduling the jobs between cloud data centers. It has been observed that the PSO variants can efficiently solve the load balancing problem and take lesser time to converge the final solution [27]. Lynn and Suganthan (2015) presented a learning based PSO technique with enhanced exploration and exploitation [28]. Two multi-agent based PSO techniques were designed and compared using different benchmark parallel problems [29, 30]. A multi-objective PSO technique using multiple search strategies were used where decomposition approach was exploited for transforming multi-objective particles into a set of aggregation problems. Then, each particle was assigned accordingly to optimize each aggregation problem [31, 32, 33].

A hybrid PSO and VNS was designed to overcome the initial number of particles issues with standard PSO technique [34, 35, 36]. Mostly, PSO was used to develop initial solutions near to optimal schedules. Based upon the gbest values, various solutions are selected for further proceedings. Then, VNS come in action to improve the results of PSO technique further [37, 38, 39]. A multi-objective fitness function was designed to reduce the energy consumption rate, makespan and load imbalance rate at a same time [40, 41, 42].

A double global optimum technique was proposed with the help of GA and PSO approaches to remove the premature convergence and convergence speed issue [43]. This technique can efficiently schedule the jobs between cloud data centers in an efficient manner [44].

To overcome the problems associated with existing scheduling techniques many researchers have proposed hybrid scheduling techniques to enhance the scheduling outcomes further [45, 46]. Arora and Singh (2013) integrated firefly, BAT and cuckoo search to find optimal solutions. Experiments showed significant results compared to standard Firefly, BAT and cuckoo search based scheduling techniques [47]. Raju et al. (2013) designed a hybrid ACO and cuckoo search based scheduling technique. Hybrid technique has reduced the makespan and energy consumption efficiently [48].

Rahmani et al. (2013) designed a hybrid ACO and PSO based metaheuristic technique to improve the convergence speed of ACO and number of particle issues [49]. Ghanbari et al. (2013) discussed a hybrid ACO-GA to reduce premature convergence issue with GA [50]. Huang et al. (2013) designed ACO and PSO based hybrid technique to minimize the load imbalancing rate of load balancing techniques [51]. Hybridization of metaheuristic techniques have improved the convergence speed [52, 53], remove the problem of premature convergence [54, 55], stuck in local optima issue [56, 57].

Most of the scheduling techniques are based on single objective issue. The multi-objecitve optimization [58, 59, 60, 61] sets more than one performance metrics as fitness function. Kachitvichyanukul et al. (2011) introduced a multi-objective two-stage job shop scheduling using genetic algorithm to solve three criteria like minimization of total weighted earliness, minimization makespan and total weighted earliness and at end also minimized total weighted tardiness for best optimization [62]. Li et al. (2012) introduced a hybrid multi objective game theory algorithm for solving the scheduling problems [63]. Shahsavari et al. (2013) introduced a novel hybrid approach to combine simulated annealing with genetic algorithm having multi objective function in load balancing problems which increase the quality of solution and reduce the computation time [64]. Wang et al. (2014) implemented a load balancing mechanism based on multi-objective GA for minimizing energy consumption and makes more profits to service providers by providing a dynamic selection mechanism [65].

Ramezani et al. (2015) considered job execution cost, job queue length, power consumption and job transfer time conflicting objectives to develop a multi-objective approach for load balancing to optimize it. Also reduced costs from both perspectives i.e. customer and provider in cloud environment using PSO and GA multi-objective features. CloudSim toolkit is also used for finding the optimal job arrangement among virtual machines [66]. Kumar et al. (2016) optimized the processing time and energy with multi-objective nested PSO in load balancing using a simulated open source cloudSim cloud computing platform [67]. Many load balancing algorithms with modified Min-Min load balancing technique in the grid system is used to enhance the Load balancing, minimizing makespan and rescheduling of jobs for efficient utilization of resources [68, 69, 12]. Joshi et al. (2016) handled multi-objective optimization with the evolutionary technique by proficient PSO to minimize computation time and fast convergence comparative to other procedures [70]. Yao et al. (2017) studied a multi-objective multi-swarm optimized algorithm to solve conflicting objectives like energy consumption, cost and makespan (total execution time) and satisfy the multiple scheduling [71].

III. MATHEMATICAL FORMULATION

Metaheuristic based load balancing techniques do not guarantee to provide optimal schedules for cloud computing environment due to their random nature. However, the results provided by metaheuristic techniques are good as compared to heuristic techniques. These techniques assign user jobs on available cloud servers in such a way that overall execution time of given set of jobs gets minimized.

PSO and GA were considered as efficient load balancing techniques for generating satisfactory results. Assuming investigation ability of GA and good convergence speed of PSO, an efficient scheduling technique has been proposed in this research work. The proposed scheduling technique utilize mutation operator of GA and implemented it in PSO to minimize the makespan, load imbalance rate and energy consumption of cloud servers.

Multi-objective fitness function is used as a minimization objective function for all designed scheduling techniques. Energy consumption, load imbalance rate and makespan parameters are used so as to design multi-objective fitness function. Job duplication is also considered to reduce the inter-communication overhead time of servers especially in case of dependent jobs. A Directed acyclic graph (DAG) G = *(*V ,E*)* compose V vertices and e edges. A V shows a decayed job, that is processed in order not considering the preemption. The execution time of job j_a 2 v on cloud (C_l) is referred as $!_{ab}$. Each e_{ab} 2 E shows precedence constraints among j_a and j_l, that signify consequence of j_a needs to be transmitted to j_l, before it initiates its execution. Each e_{ab} 2 E is of a non-negative weight (nw_{ab}) showing the inter-communication cost consisting interdependent j_a and j_l. The actual communication weight corresponds to 0 once the co-dependent set of jobs are assigned to the identical servers. Suppose that, a job grid with M jobs is to be assigned to a cloud model with nop servers. Given an incomplete optimistic job allocation matrix, taking into account the optimistic job allocation, the highest-priority prepared job j_a on the cloud C_l, its initial start time $S_t(j_a; C_l)$ may be defined as follows:

$$S_t(j_a; C_l) = \max\ S_{ae}(C_l); S_r(j_a; C_l) \quad (1)$$

Here, $S_{ae}(C_l)$ represents the time when cloud (C_l) is available to process the job (j_a). It is computed as follow:

$$S_{ae}(C_l) = \max\ ex(C_l); S_{ft}(j_f) \quad (2)$$

Here, $ex(C_l)$ shows set composing jobs which have been recently assigned on C_l while $S_{ft}(j_f)$ shows actual execution time when j_f ends up its execution. $S_r(j_a; C_l)$ shows the time when all information required for the execution of j_a is communicated to C_l and can be computed as follows:

$$S_r(j_a; C_l) = \max\ ex(j_a); S_{ft}(j_f) + nw_{f;i} \quad (3)$$

Here, $ex(j_a)$ shows a set compose of job (j_f). The $nw_{f;i}$ is assigned to be 0, if j_f has been assigned to C_l. Assuming that j_a is scheduled on C_l with non-preemption technique; its actual finish time ((S_{ef} (j_a; C_l))) is computed as follows:

$$S_{ef}(j_a; C_l) = S_t(j_a; C_l) + !_{ab} \quad (4)$$

Here, $!_{ab}$ shows the execution time of j_a on C_l. $S_t(j_a; C_l)$ and S_{ef} (j_a; C_l) are allocated to $S_r(j_a)$ and $S_{ft}(j_a)$, respectively. Also, termination vertice (j_{EXIT}) represents the ms.

$$ms = S_{ft}(j_{EXIT}) \quad (5)$$

IV. PROPOSED TECHNIQUE

To handle the issue of load balancing in cloud computing environment, a typical cloud model is designed. Cloud model contains several geologically distributed high end servers associated using internet. Prin-cipally high-end servers contain of numerous computing and storing resources. These high-end servers communicated with each other using a high bandwidth intercommunication network. Thus, in the de-signed cloud model, transmission delay does not play a significant role. In designed cloud environment, each user can utilize cloud resources with the help of internet. Cloud service provider is responsible for allocation or deallocating the resources to users. The user jobs are disseminated between several cloud data centers . Each decompose user job into sub-jobs so called jobs and allocate it between (DC_s) DC_s available processing elementsin the respective. The designed load balancing technique ($C_{ro}E_S$) (DC_s) is responsible for efficiently assigning of user jobs into available with an objective to reduce the (DC_s) makespan time and average waiting time.

Assume that Job = (J_{S1}; J_{S2}; J_{S3}:::J_{SM}) is a group of applications received in a specific period of time from M users. Every job (J_{Sj}) is considered by a duplet < A^0_i; D_i >. In which A^0_i defines arrival time of job (J_{Sj}) and D_j represents deadline of job (D_j) If a job could not finish within deadline time, then it is referred as a failed job and queued again for further processing. Throughout the scheduling procedure,

jobs are allocated to data centers (DC_s)(D_1 ; D_2 ; D_3:::D_N). Here, N M. Each Dj is associated with a duplet < c_j; n_j > :c_j; cost per unit time charged by DC $_s^0$ to implement jobs, n_j is the number of available processing elements (PE) to implement jobs. Each DC^0 have set of PE C r E; C r E 2:: C E to s s f o 1 o : ro ng

Evaluate assigned job. Each PE is associated with a duplet < s; p >. Where, s and p represent the burst time and energy consumption of each to evaluate assigned job. Every Job is demonstrated as a Directed acyclic graph (DAG), represented as g(v; E). The set of nodes= fj$_1$:::; j$_m$ g shows jobs and the set of arcs represents the data dependencies among jobs. An arc is in the form of <fj$_i$:::; j$_j$g >2 E , where j_j represent parent job and j_j is a leaf job. j_j cannot be implemented until all of its root jobs have been

implemented. Assume that user job is allocated to data center . define set of jobs allocated to

J_j D_J j_A J_j

a PE(C_j). If the time demands executing j_A using C_i is represented by $_j$. The deadline time of j_i can be evaluated as follows:

$$finish(\ _j) = start(j_A) +\ _j \quad (6)$$

Therefore, burst time required to finish the job by D_i is rerpresented by Makespan MS_i and calculated as follows:

$$MS_i = \max ffinish(j_A)g \quad (7)$$

Here, $j_{(A=1:::n)}$ is the jobs that are assign to D_i. The Energy consumption (E_j) to evaluate a job (J_j) is evaluated as follows:

$$E_j = S_A{}^m{}_{=1}(\ _A\ \ C_A) \quad (8)$$

Here, C_A represent power consumed per unit time by PE (C_i) to execute given job (j_A). The cost to

execute the job by is evaluated as follows:
D_i

$$c_j = C_j \quad MS_j \tag{9}$$

Here, c_j is the price per unit time charged by D_i to implement job. The utilization U_j of D_i is evaluated as follows:

$$MS$$
$$\max f MS_k g \tag{10}$$

The fitness functions of this proposed model can be represented as follows:

$$\text{Minimize} MS_j j = 1::N \tag{11}$$
$$\text{Minimize} E_i i = 1::N \tag{12}$$
$$\text{Minimize} C = S_{t=1}^{n} c_j \tag{13}$$
$$\text{Minimize} J_i i = 1::N \tag{14}$$

Subject to: 1. The job must finish before deadline (D_i)

2. Every job can be assigned to only one .
D_i

3. Number of jobs must be less than the number of available Data.

In this experiment, a mutation based PSO has been designed. A step by step methodology is used. Initially, the cloud based model is designed. The DAG is designed for the FFT problem. Then proposed technique will be formulated. In proposed technique, first of all, random initialization of given set of solutions will be done. Then, PSO comes in action to evaluate the optimal load balanced schedule. It can evaluate the global optimal solution. However, PSO is limited to the initial set of particles, which means wrongly selected particles may lead to poor results. To overcome this issue, the proposed technique will end up by optimizing the solution with the mutation operator technology. Thus, proposed technique will have an ability to find the optimal solution in more optimistic manner. The subsequent section contains the detail of each technique with suitable procedures and required formulas. Initially, a set of random particles are developed. Set of group particles were developed with the help of encoding technique and a solution was obtained from the every particle in population set. Particles list constructed with the help of integer numbers randomly. The random integer numbers are created by permutation technique. The encoding procedure of particles having high priority job list to low priority job list. Here, n genes or jobs or sub-jobs (can be taken as sub-jobs also) executed or scheduled in a priority list (high order priority to low order priority) or in a sequence. Topological order is the best for arrangement of subjobs or jobs. Therefore, job or subjobs are evaluated as these occur having DAG with initial population size C_{size} = 4. Further, top level rank (Tr), bottom level rank (Br) and top-bottom level rank (T-Br) are the three major heuristic rank techniques used in this design. To show a better priority queue list (PQL) a better seeding procedure must be adopted for initial population and to obtain this a three level ranking was illustrated well in Eqs. (4.28), (4.29) and (4.30) respectively [72]. This is the HEFT based PQL. A PQL values were shown in Table 1. Here, CP taken as the position of particles.

Table 1: Priority queue list showing the job priority queue

CP	1	2	3	4	5	6	7	8	9
Br PQ	j_3	j_4	j_7	j_1	j_6	j_2	j_9	j_5	j_8
Tr PQ	j_3	j_2	j_1	j_8	j_4	j_6	j_5	j_7	j_9
TBr PQ	j_9	j_8	j_7	j_6	j_5	j_2	j_1	j_3	j_4

$$rank_{top}(j_a) = \overline{ACC(j_a)} + \max_{j_s 2 succ(j_a)} (QC(j_a; j_s) + rank_{top}(j_s)) \tag{15}$$

Here, the $ACC(j_a)$ is the average computational cost of subjob w_a, $QC(j_a; j_s)$ is quantity of commu-nication between the subjobs j_a and j_s and $rank_{top}(j_s)$ is the upward rank of subjob j_a's successor.

$$rank_{bottom}(j_a) = \max_{j_s 2 pred(j_a)} rank_{top}(j_s) + \overline{ACC(j_s)} + QC(j_a; j_s) \tag{16}$$

Here, $rank_{bottom}(j_a)$ is the downward rank of the subjob j_a's precedence.

$$rank_{top\ bottom}(j_s) = rank_{top}(j_a) + rank_{bottom}(j_a) \tag{17}$$

The processing of initial population is described as follows:

Algorithm 1 : Initialization or initial population creation

1) Take population size C_{size} and particle size C_{size} as input values
2) Set i = 3
3) Initializing i particles set using three heuristic rank techniques
4) Generate population randomly after initialization of particle set
5) For j = 1 to (j C_{size} 1)
6) For k = 0 to (k C_{size} 1)
7) Create a new particles k values randomly
8) Now particle i position changed from left to right in a queue
9) Endfor
10) Endfor
11) Stop the processing of creation of genes

A. Assignment Of Sub-Jobs To High-End Machines

In the case of originated population, every individual should have a major priority mechanism having permutation process. Therefore, subjobs should follow precedence conditions for this process. A subjob will be allocated to the server with maximum speed, if and only if it is not already scheduled. In the case of proposed approach, HEFT technique is utilized to define the subjobs with maximum priority in the individuals. Further, it allocates given subjobs to the server(s) in such a way that it minimize the overall ms.

The initial start time (I_{ST}) of the subjob j_a on processor C_i is symbolized as I_{ST} (j_a; pi) which is obtained as follows:

$$I_{ST}(j_e; C_i) = 0 \quad (18)$$

$$^{I}ST\ ^{(j}a;\ ^{C}i)\ =\ ^{max}j_s\ pred(j_s)\ +\ ^{A}ST\ ^{(j}s)\ +\ (C(j_a; j_s)) \quad (19)$$

Here, j_e is job entry. The actual start of sub job j_a on processor C_i is symbolized as A_{ST} (Actual start time) (j_a; C_i). This is computed as follows:

$$A_{ST}(j_a; C_i) = \max(I_{ST}((j_a; C_i); avail(C_i)) \quad (20)$$

Here, avail(C_i) is time that the processor C_i has idle and ready for the job execution. The earliest finish time of subjob j_a on processor C_i is symbolized as I_{FT} (j_a; C_i) which is obtained as follows.

$$I_{FT}(j_a; C_i) = CC(j_a; C_i) + A_{ST}(j_a; C_i) \quad (21)$$

Here, CC(j_a; C_i) is the computational cost of the subjob j_a on processor C_i . The actual finish time A_{FT} is computed as follows.

$$A_{FT}(j_a; C_i) = \min E_{FT}(j_a; C_i) \quad (22)$$

1 l P

Allocation of jobs or subjobs to n servers or using the load balancing criteria is a significant achieve-ment in proposed technique and procedure is described as follows.

Algorithm 2 : Job allocation

1) Initialize current population values
2) Input the particle size value as C_{size}
3) Evaluate schedule length or makespan ms from a priority queue list of jobs or subjobs allocated PQL
4) While (PQL 6= Null) do
5) First job or subjob from PQL is selected
6) For processors pi = 1 to n
7) Evaluate Ft or (ms) as fitness function using HEFT scheduling process
8) If rand JDR, then assign i^{th} subjob to all virtual machines and evaluate maximum schedule length or makespan (ms), Else, Allocate i^{th} subjob to j^{th} servers and evaluate maximum schedule length or makespan (ms)
9) Assign jobs or subjobs to virtual machines
10) Evaluate ms = max(schedule length)
11) Endfor
12) Remove i^{th} subjob or job from PQL
13) End while loop
14) Return the final value of makespan ms

B. Mutation Operator

There are two types of mutation in GA, naming immigration and allele type. In this algorithm immigration mutation is used. For mutation process, a mutation operator is used to process a diversity in particle population with a defined mutation probability rate. Here, a particular particle is replaced randomly with another particle and evaluate the fitness function makespan ms till a better optimize result arrived. This process is shown in Figure 1. An iterative process of mutation will occur until a better optimal solution will be obtained. This process is illustrated in Algorithm 13.

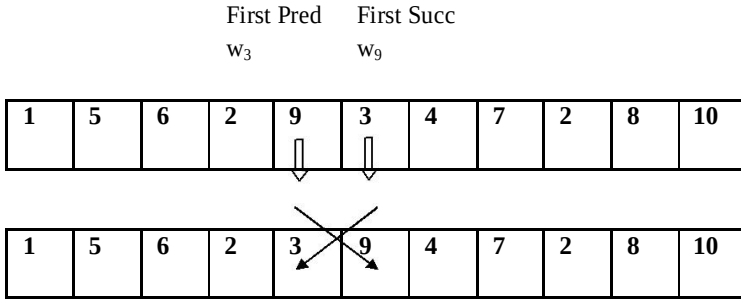


Figure 1: Mutation operator

Algorithm 3: Mutation operation process

1) Inputting the particle list of current population
2) A new population generated by applying mutation operator
3) Select the particle list randomly and Let G = rand(gene)
4) Select the first successor and Let S = rand(gene) from particle list except G
5) Offspring= interchanging G with S
6) Evaluate the makespan ms
7) If ms=minimum then terminate else go to the first step again (i.e. step 1).
8) Print optimal result having makespan ms

C. Termination Condition

Since, it is not possible to achieve ms as 0, therefore, 100 function evaluations used to stop each section of the algorithm. The function evaluations mean total number of time tried to calculate the ms, so called fitness function or objective function.

D. Particle Swarm Optimization

The optimistic schedule generated by the GA, will act as initial number of particles for the PSO. This technique can be called as guided GA based PSO.

Thus, it can overcome the issue of "poorly selected particles tends to poor results" with the standard PSO. Let us take C_i which represent i^{th} particle at a particular iteration. It has n number of dimensions and be represented as: C_i= [C_{i1}, C_{i2}, C_{i3}... C_{in}], where C_{ij} denotes position value of i^{th} particle w.r.t. j^{th} dimension. PoC_i in the swarm optimization have r set of particles operated at i^{th} iteration and is represented as: PoC_i= [C_1; C_2; C_3; C_4; :::C_r]. PV_k represents the velocity of k^{th} particle moving in a particular iteration. Eq. (4.24) is represented as: PV_k= [pv_{k1}; pv_{k2}; pv_{k3}:::pv_{km}], Here, pv_{ij} denotes the velocity of k^{th} particle w.r.t. d^{th} dimension.

In load balancing, permutation (Permu) helps in generating and searching the particles path. It is represented by $PermJ_i$. It is applied on particles as C_i:$PermJ_i$= [$permJ_{i1}$; $permJ_{i2}$, $permJ_{i3}$, ...
$permJ_{in}$]. Here, $permJ_{ij}$ represents assignment of job j of the particle i using the permutation $PermJ_i$ in particular iteration w.r.t. j^{th} dimension. Computed weight or inertia weight or inertia (IW) is a powerful attribute and is used to control the impact of previous velocity on the current velocity.

Personal best (pbest or PB) shows best position of i^{th} particle having best fitness till j^{th} iteration. PB for every particle can be computed in each iteration. Further it can be updated after every iteration.

Global best position (gbest or GB) from all the sets of local particles. It is the best from whole swarm and gives the better optimization solution. Machine Sequence ($mseq_i$) represents machine se-quence of k^{th} particle. It occurs at particular sequence and iteration and is: $mseq_k$=[$mseq_{k1}$, $mseq_{k2}$, $mseq_{k3}$,...$mseq_{km}$], Here, machine sequence is $mseq_{k1}$. It works for k^{th} particle w. r. t. i^{th} dimension.

PSO for evaluating the load balancing to get the optimized solution is well elaborated in Algorithm

E. Applying Mutation For Optimized Result

To generate heuristic based local search, select the suitable neighborhoods. The efficiency of mutation operator increases the correct choice of neighborhoods. Mostly three neighborhoods used in getting the better efficiency. One is having exponential size and other two have polynomial size. In load balanc-ing, the assignment is the biggest problem and optimal solutions can be obtained with the use of large neighborhood elements. Efficiency can be increased by doing so.

Let us take x = FS as a feasible solution. Taking Move(x) operation that represents feasible solutions set, can be attained from x by selecting one job at a time, removing the job from the server. Further, assign the removed job into another server. It will create a relationship of the server with the jobs and such neighborhood has most jDj jSj solutions. All the feasible solutions can be obtained from second neighborhood denoted as Swap(x). It can be done from x by exchanging two jobs between different servers. It consists of at most D^2 solutions. Randomized versions of neighborhoods can be used due to polynomial nature of neighborhoods. So, efficiency in local search can be increased that it will also minimize the computational cost per iteration.

Let q is the parameter used in randomized neighborhood having operation Moveq(x). Here, q range is from 0 < q < 1. Moveq(x) is a element of neighborhood Move(x). Every part of Move(x) is inclusiv

Algorithm 4: To get the optimized solution using particle swarm optimization

1) Intialization: Initialize the population sets of swarm and set i = 0 and j= best solution by particles. Each particle's position and its velocity are taken for evaluating the best schedule.

2) Apply smallest position value rule (SPV) and its operation: There is a SPV applied on the set of jobs to get the permutations of jobs for every particle to get the better solution as compared to previous procedures.

3) Evaluation of ms: The evaluation of ms for every permutation depends upon the representation of jobs. Note that personal best (pbest) can be computed from the ms evaluation. So, in every iteration, the minimum ms of a particle is the best position of that particle.

4) Let i = i + 1 counter work for new iteration and upgrade the iw as follows:

$$= iw^{i\ 1}:x \text{ ; (Here, is the decrement factor)} \qquad (23)$$

5) Upgrade the velocity as follows:

$$PV_{kj}^{\ i} = iw^{i} + PV_{kj}^{\ i\ 1} + c1:j1(C_{kj}^{\ i\ 1} \quad PB_{kj}^{\ i\ 1}) + c2:j2(C_{kj}^{\ i\ 1} \quad GB_{kj}^{\ i\ 1}) \qquad (24)$$

Here, c1 is social parameter and c2 is cognitive parameter. between (0; 1).

6) Upgrade the position as:

$$C_{kj}^{\ i} = C_{kj}^{\ i\ 1} + PV_{kj}^{\ i} \qquad (25)$$

j1 and j2 are random numbers

7) Changing sequence of jobs: With the help of the SPV rule, the particle's position can be updated by applying permutation operation on jobs. It will change the sequence of jobs.

8) Update of pBest: By comparing the old one pBest with new one personal, the pBest can be upgraded. If the new one has the lower value as compared to old one, then the new one is best and so updated with old one.

9) Evaluate gBest: gBest can be evaluated from the set of computed pBest. The pBest having minimum value from all the pBest is the gBest.

10) Termination criteria: Procedure can be stopped if the counter of iteration exceeds the maximum value of iteration (It is also a comparison of maximum utilization of CPU time). into Moveq(x) having probability q. Similarly, neighborhood Swapq(x)is defined.

Also, x represents neighborhood function and is used for the construction of elements by using server and jobs. For every server, a set of jobs Ds is defined. Here, s 2 S be removed from server S. Further, remove one job from every set Ds. D is denoted as the subset of removed jobs. In the case of Ds = 1 occurs for the server s, then remove a dummy job.

The primary purpose of this operation is to enhance the performance of all servers in a particular time interval. The given condition ensures the job movement i.e. every job has movement only to one server and so every server has exactly one job at a time.

General algorithm for mutation operator is described as follows:

Algorithm 5: To get the better-optimized results using mutation operator

1) Assign SOL = FS

2) Following steps be repeated till stopping criteria met.

3) Set i := 1;

4) Do while i = imax:

5) Select i jobs from optimistic solution developed by PSO procedure
6) Move these jobs to other servers
7) By applying local enhancement procedure on neighborhood Moveq, Swapq.
8) By applying the assignment operation to improve the current solution lmax times.
9) If improved solution x1 is better than x, then x := x1 and i := 1, else, i := i + 1.
10) End of while loop.
11) Return x.

In this algorithm, execution time can be computed as soon as stopping criteria met. Select the ele-ments from sets Ds randomly having the uniform distribution.

The proposed technique utilizes four updating velocity mechanisms to efficiently get away from non-global optima and develop a result quality [24]. Cloud computing environment has a high-end area of cloud, all of that includes a confined memory to ensure that inter-cloud interactions depend exclusively on frames exchangement. The processing weight of a precise job j_a " v on cloud (C_l) is referred as $!_{ab}$. Each edge e_{ab} " E represents precedence constraints between jobs j_a and j_l, which signify consequence of job j_a needs be passed on to job j_l prior to job j_l initiate execution [9]. Every edge e_{ab} " E is of a non-negative weight nw_{ab} showing communication cost involving interdependent jobs j_a and j_l. The actual message passing weight is corresponding to zero once the co-dependent set of jobs is allocated to the identical cloud. The originated vertice of a connection is known as the ancestor and the terminated vertice is known as the descendant. The job having no parent is reffered as the relay job where as job without any leaf child is called the terimination job [9]. Suppose that, a job grid with M jobs are to be assigned to clouds. Given an incomplete optimistic job allocation matrix, taking into account the optimistic job allocation the highest-priority prepared job j_a on the cloud C_l, its initial start time S_t (j_a,C_l) may be defined as:

$$S_t(j_a; C_l) = \text{maximum } fS_{ae}(C_l); S_r(j_a; C_l)g \quad (26)$$

Here, $S_{ae}(C_l)$ referred as the moment at what time cloud C_l is existing to the processing of the job j_a. It is referred as:

$$^{S}ae^{(C}l^{) = \max}tf''ex(C_l)\ ^{fS}FT\ ^{(j}f\ ^{)g} \quad (27)$$

Here, $e_x(C_l)$ represents the group containing all jobs which have recently been optimistic job allocation on the cloud C_l while S_{FT} (j_f) symbolizes the real completion moment when the job j_f in reality termination of its processing. Furthermore, S_t (j_a, C_l) in Eq. (4.39) indicate the moment when entire information required for the processing of job j_a that is broadcasted to the cloud C_l, that can be referred as follows:

$$^{S}R^{(j}a^{; C}l^{) = \max}tf''pr(j_a)\ f^{S}FT\ ^{(j}f\ ^{) + nw}f;ig \quad (28)$$

Here, pr(j_a) represents the group containing all instantaneous forerunner of the job (j_f), the $nw_{f;i}$ set to be 0 if the job (j_f) has been allocated to the similar cloud (C_l). Assume that job (j_a) is scheduled on the cloud (C_l) with non-preemption execution technique; its original termination time. S_{EF} (j_a,C_l) can be defined as follows:

$$S_{EF}\ (j_a; C_l) = S_t(j_a; C_l) + !_{a;b} \quad (29)$$

Here, $!_{a;b}$ symbolize the processing time of the job j_a on the cloud C_l. Following the job j_a is unequiv-ocally allocated on the cloud C_l, the S_t (j_a,C_l) and S_{EF} (j_a,C_l) are assigned to S_t (j_a) and S_{FT} (j_a) correspondingly. In general ms of the complete concurrent process, namely turnaround time, is the max-imum time from given set of jobs that is equal to the real turnaround time of the termination vertice
$^{(j}EXIT\ ^{).}$

$$ms = \max_{ta''v}\ fS_{FT}\ (j_a)g = S_{FT}\ (j_{exit}) \quad (30)$$

Here, ms represents makespan. Assume that V M_i ($_1$; $_2$; ; $_n$) is a set of devices of CCE. Let $_1$ is the local distributed server and $_2$; ; $_n$ are external clouds (ECs). = $_1$, $_2$; ; $_I$ is a set of Processing element (PE) and = $_1$, $_2$; ; $_w$ is a set of jobs. Every job ($_j$) (j f1; 2; ; wg has a predefine finishing time $%_j$ and computation time r_j and also compose of a job set J_j= fJ_{j1}; J_{j2}; ; J_{j3}g. Let TS be the given number of time intervals and also TS = $\max_{j\ f1;2;\ ;wg}$ ($%_j$). Major goal by assigning w jobs to $_k$ k (1; 2; ; n) to capitalize on gain of $_1$. Every job would be assigned to one $_k$ k (1; 2; ; n).

One time job is initiated for computing, it cannot be preempted, therefore computation intervals are consecutive. At any interval TS (TS (1; 2; ; S)), equipments utilized by all jobs computed in $_1$ can never acquire the maximum number of equipments of $_1$. The gain assumption is as:

Maximize

$$\text{Gain} = S_j^{a}{}_{=1}S_v^{I}{}_{=1}J_jb_{jv}C_v \quad S_j^{a}{}_{=1}S_I^{J}{}_{=1}{}^{j}S_v^{1}{}_{=1}S_k^{n}{}_{=1}y_{jIK}\ b_{jv}c_{kv}r_j \quad (31)$$

Subject to:

$$S_k^{n}{}_{=1}y_{jIk} = 1;\ 8j''\ (1; 2:::J_j)$$

(32)

$$S_{TS}^{\%j}{}_{=1}Z_{jIS} = y_{jI1}r_j; \; ; \; 8j"(1; 2; ::::w); \; I \; (1; 2; :::J_j) \quad (33)$$

$$TSJ_{j1} \quad 1; \; 8j"(1; 2; ::w); \; 1"(1; 2; ::::J_j) \quad (34)$$

$$TSJ_{j1} \quad \%_j \quad r_j + 1; \; 8j"(1; 2; ::::w); \; 1"(1; 2; ::::J_j) \quad (35)$$

$$(TS \quad TSJ_{jI}) \_ (TS \quad \%_j \quad r_j) \_ ((TS \quad TSJ_{jI}) \wedge (TS \quad TSJ_{jI} + r_j)) \quad (36)$$

$$S_j^a{}_{=1}S_I^J{}_{=1}{}^jS_v^I{}_{=1}Z_{jIs}b_{jv}CPJ_v \quad J_{CPU} \; ; \; 8TS"(1; 2:::TS) \quad (37)$$

$$S_j^a{}_{=1}S_I^J{}_{=1}{}^jS_v^I{}_{=1}Z_{jIs}b_{jv}MES_v \quad J_{MEM} \; ; \; 8TS"(1; 2:::TS) \quad (38)$$

$$Y_{jIk}" \text{ range } 0 \text{ to } 1; \; 8j"(1; 2; :::a); \; "(1; 2; ::::J_j); \; k"(1; 2; ::::n) \quad (39)$$

$$Z_{jITS}"(0:::::1); \; 8j"(1; 2; ::::a); \; I"(1; 2; ::::J_j); \; TS"(1; 2; :::::TS) \quad (40)$$

$$TSJ_{j1}"(1; 2; ::::TS); \; 8j"(1; 2; ::::a); \; I \; 2 \; to(1; 2; :::::J_j) \quad (41)$$

The objective function given in Eq. (4.44) has clearly shown that the earning of $_1$ and the follow-ing is its cost and condition Eq. (4.45) guarantees that each job is assigned to accurate cloud servers. Condition Eq. (4.46) defines that each job is accomplished before its maximum finishing execution time. Conditions Eqs. (4.46) to (4.48) guarantees that each job is non-preemptable. Conditions Eqs. (4.49) to Eq. (4.51) are assigned to $_1$ to define that it will not take CPUs and memory more than its predefined capacity, in each interval. At last Eqs. (4.52) to (4.54) give definitions of the various constraint variables and constants.

The proposed technique initializes with random particles on the basis of a random variable theory. After following the job ranking, the particle is developed and prioritized with maximum turnaround time being the objective of every particle. Some of genetic characteristics consisting flipping mutation, crossover are then applied on developed particle. After that, the particle is classified to able to choose a sub-group containing of an individual for particles. The non-global exploration domain process of PSO is placed on every character in the chosen sub-group.

F. Stopping Method

The proposed technique technique will return final results in two cases, either the defined number of iterations met or the fitness remain constant for more than ten iterations.

V. COMPARATIVE ANALYSIS

This section illustrates the experimental set-up and quantitative analysis of the proposed load balancing techniques. The proposed techniques have been tested on well-known Fast Fourier transform (FFT) benchmark parallel problem. Different sizes of jobs have been considered to evaluate the scalability effect of the proposed techniques. All load balancing was executed on Intel core i5 processor @ 2:56 *GHz* with 16 GB RAM. MATLAB 2013a software is used in combination with parallel processing toolbox. In this section, performance of proposed technique has been evaluated and compared with GA, ABC and PSO based load balancing techniques. Subsequent section compares the proposed technique with existing load balancing techniques based upon some well-known performance metrics.

Table 2 and Figure 2 depicts the comparison between proposed technique, GA, ABC and PSO in terms of makespan (ms). It has been found that the proposed technique has lesser ms compared to existing techniques.

Table 2: Makespan analysis of proposed technique

V M_i	FFT	GA	ABC	PSO	Proposed
2	4	11212 127	11015 106	10257 79	10045 61
	8	16799 194	16081 181	15924 147	15398 124
	16	24132 287	23571 193	22022 193	21419 169
4	4	9255 234	9144 235	8285 198	4693 173
	8	12000 291	11103 287	9686 249	9353 204
	16	14200 287	13756 312	11385 276	10858 219
8	4	9287 274	9188 194	7848 174	7427 136
	8	10200 319	9983 177	7942 194	7556 163
	16	8799 227	8544 203	7352 186	7139 159

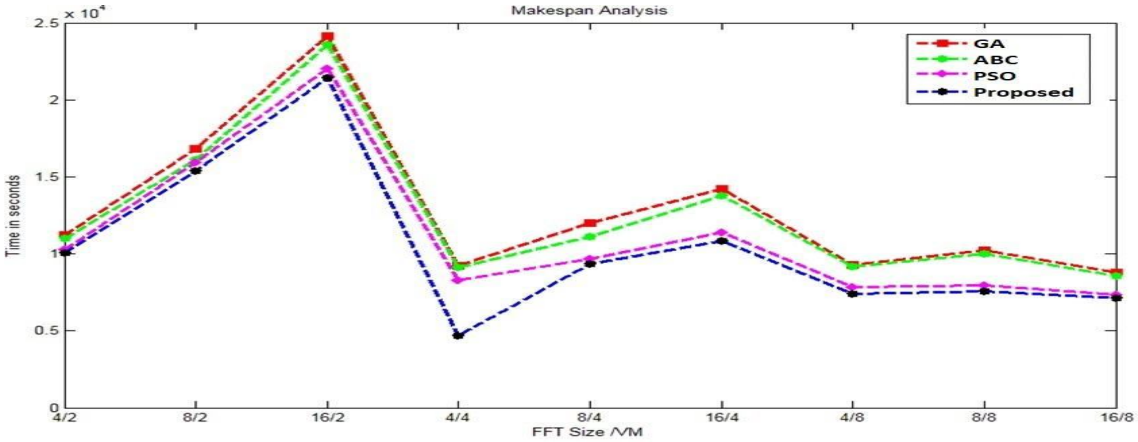


Figure 2: Makespan analysis of proposed technique

The comparison between proposed technique, GA, ABC and PSO based load balancing techniques by considering the speedup (sp) has been shown in Table 3 and Figure 3. It has been demonstrated that proposed technique technique has significant sp value as compared to other load balancing techniques.

Table 3: Speedup Analysis of proposed technique

V M_i	FFT	GA	ABC	PSO	Proposed
2	4	1.72 0:13	1.76 0:13	1.89 0:11	1.93 0:10
	8	1.98 0:09	2.03 0:07	2.05 0:08	2.12 0:08
	16	1.87 0:12	1.99 0:09	2.13 0:07	2.21 0:07
4	4	2.03 0:10	2.12 0:13	2.04 0:11	2.16 0:07
	8	2.16 0:09	2.19 0:17	2.17 0:13	2.32 0:09
	16	2.22 0:12	2.21 0:12	2.12 0:12	2.29 0:06
8	4	2.02 0:19	2.01 0:14	2.17 0:11	2.21 0:09
	8	2.07 0:16	2.07 0:13	2.11 0:13	2.32 0:10
	16	2.11 0:19	2.09 0:14	2.18 0:19	2.27 0:07

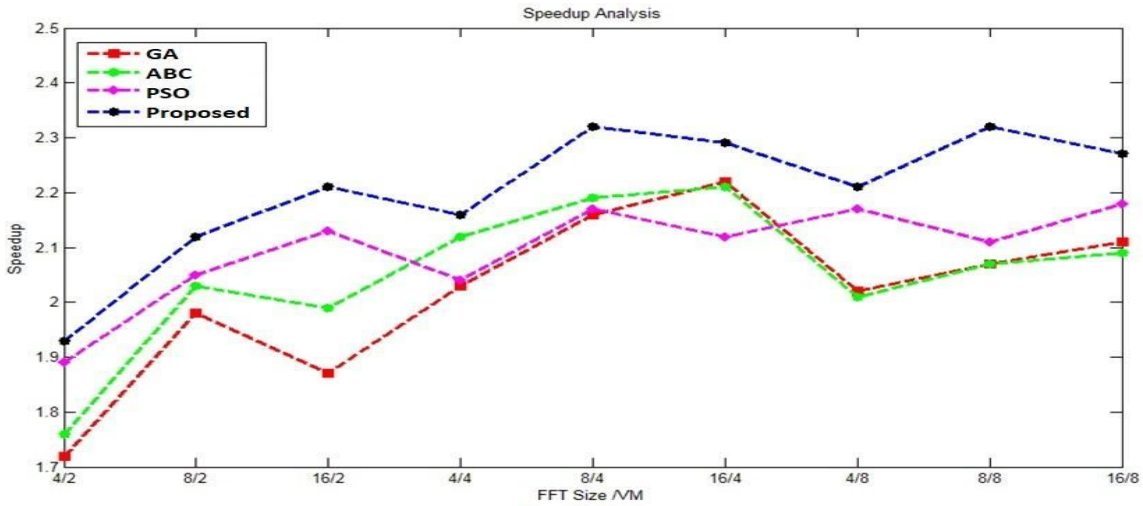


Figure 3: Speedup analysis of proposed technique

Table 4 and Figure 4 depict a comparative analysis of proposed technique with existing load balancing techniques i.e., GA, ABC and PSO by considering the Efficiency (E_f). It has been observed that the proposed technique has significant improvement in terms of E_f than existing load balancing techniques.

Table 4: Efficiency Analysis of proposed technique

V M_i	FFT	GA	ABC	PSO	Proposed
2	4	0.82 0:057	0.85 0:057	0.87 0:053	0.91 0:051
	8	0.81 0:062	0.84 0:048	0.88 0:043	0.93 0:031
	16	0.79 0:068	0.81 0:043	0.89 0:039	0.95 0:027
4	4	0.48 0:278	0.53 0:217	0.58 0:194	0.73 0:094
	8	0.67 0:142	0.73 0:137	0.74 0:125	0.87 0:071
	16	0.69 0:136	0.78 0:128	0.83 0:079	0.89 0:039
8	4	0.23 0:291	0.26 0:215	0.30 0:206	0.42 0:134
	8	0.37 0:247	0.40 0:195	0.51 0:141	0.64 0:091
	16	0.61 0:194	0.68 0:148	0.79 0:092	0.82 0:086

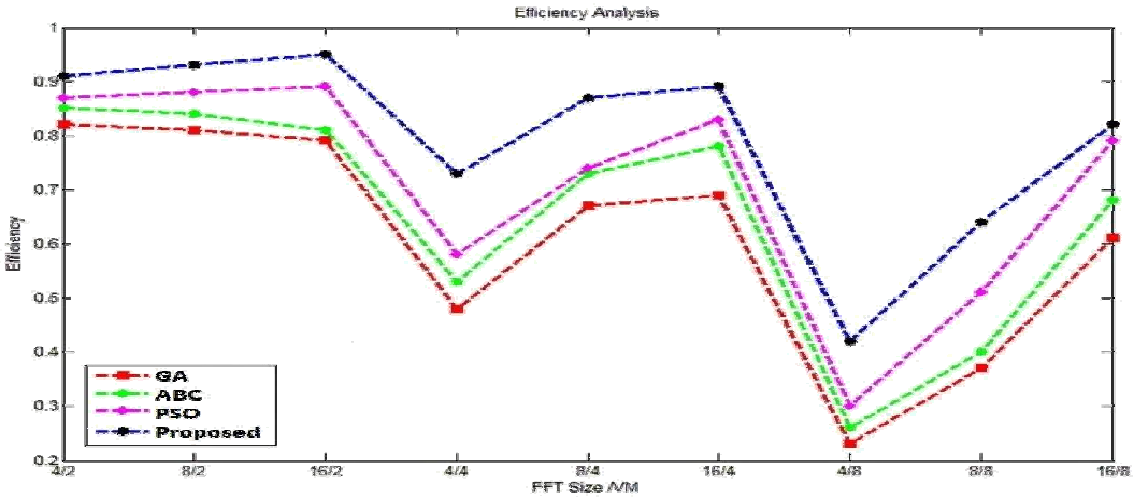


Figure 4: Efficiency analysis of proposed technique

The comparison between proposed technique, GA, ABC and PSO has been demonstrated in Table 5 and Figure 5 on the basis of J_t analysis. It has been observed that proposedtechnique has significant improvement over other load balancing technique on the basis of J_t.

Table 5: Utilization Analysis of proposed technique

V M_i	FFT	GA	ABC	PSO	Proposed
2	4	0.64 0:418	0.71 0:398	0.74 0:296	0.87 0:178
	8	0.76 0:259	0.79 0:341	0.82 0:241	0.89 0:154
	16	0.79 0:241	0.82 0:194	0.86 0:214	0.90 0:047
4	4	0.41 0:214	0.45 0:209	0.48 0:182	0.54 0:134
	8	0.64 0:197	0.69 0:189	0.74 0:137	0.78 0:096
	16	0.78 0:083	0.85 0:073	0.91 0:068	0.94 0:036
8	4	0.16 0:319	0.18 0:301	0.20 0:194	0.33 0:192
	8	0.32 0:213	0.38 0:191	0.41 0:147	0.54 0:107
	16	0.58 0:165	0.64 0:116	0.69 0:116	0.79 0:064

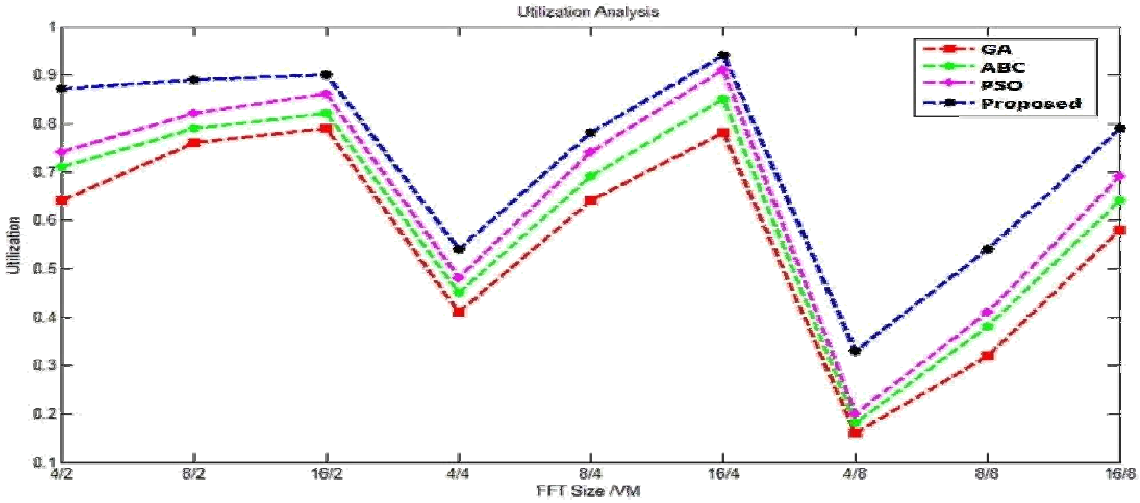


Figure 5: Utilization analysis of proposed technique

Therefore, experimental results have shown that the proposed technique based load balancing tech-nique outperforms other techniques. Therefore, the proposed technique technique is more suitable for real-time cloud computing environment to provide high availability to cloud users.

VI. CONCLUSION

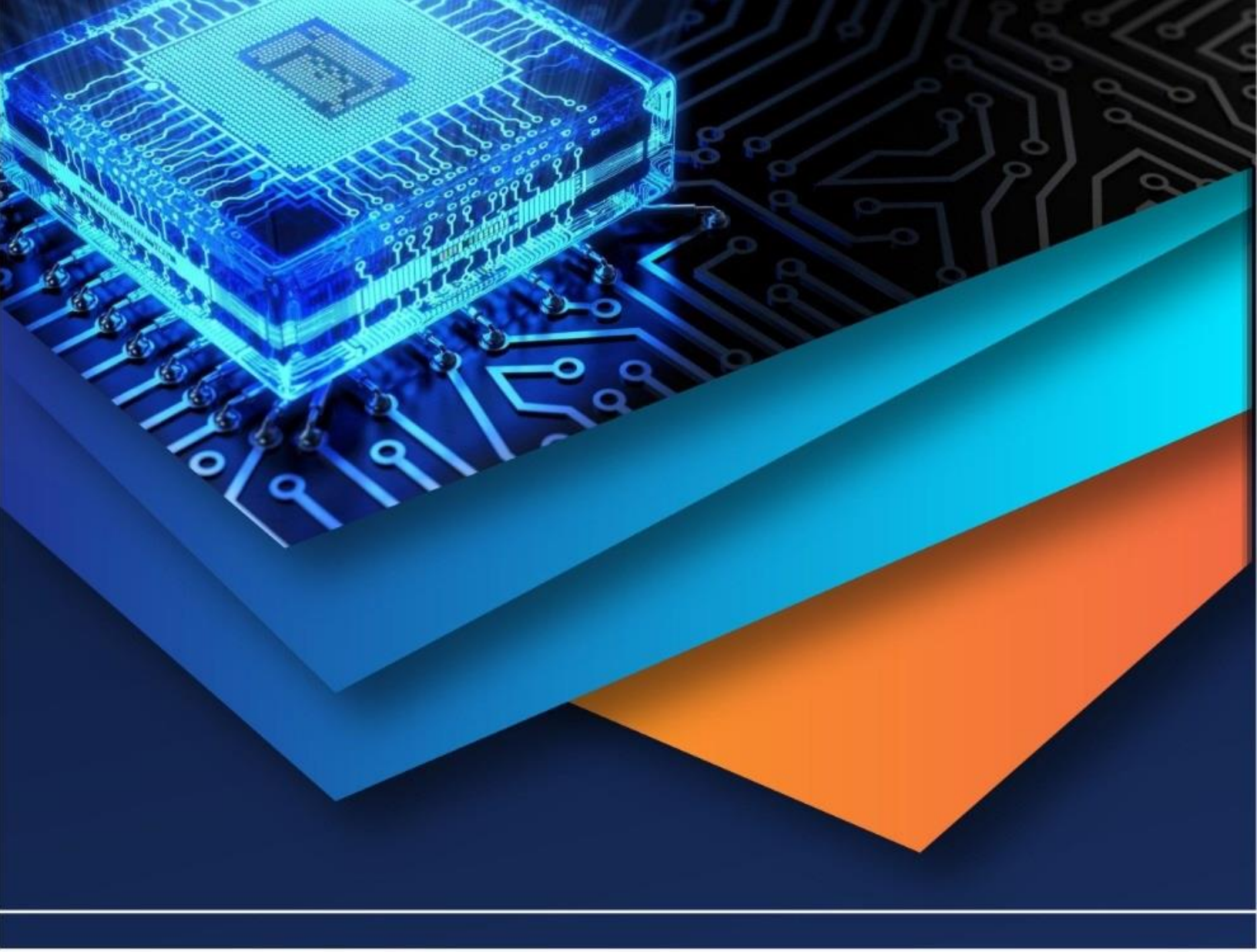
Majority of the existing metaheuristic techniques suffer from one of these issues: premature conver-gence, poor convergence speed, initially selected random solutions and stuck in local optima. Also, ma-jority of existing techniques have focused on the scheduling of independent jobs only. The effectiveness of proposed technique has been evaluated by comparing the performance of proposed technique with GA, ABC and PSO based load balancing techniques. It has been observed that the proposed technique improved speedup, efficiency and utilization by 1:74%, 0:87%, and 1:20%, respectively than GA. The pro-posed technique reduced makespan, energy consumption and load imbalance rate by 8:76%, 1:46% and 3:25%, respectively than GA. Compared to PSO it has been evaluated that proposed technique improved speedup, efficiency and utilization by 1:47%, 0:78% and 1:10% respectively. Also, it has been shown that proposed technique reduced makespan, energy consumption and load imbalance rate by 7:46%, 1:32% and 3:06% respectively. Comparisons between proposed technique and ABC showed that proposed tech-nique improved speedup, efficiency and utilization by 1:20%, 0:60% and 0:86% respectively and reduced makespan, energy consumption and load imbalance rate by 6:17%, 1:27% and 2:83% respectively.

REFERENCES

[1] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, “Cloud computing: Distributed internet computing for it and scientific research,” IEEE Internet computing, vol. 13, no. 5, 2009.
[2] Y. C. Lee and A. Y. Zomaya, “Energy conscious scheduling for distributed computing systems under different operating conditions,” IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 8, pp. 1374–1381, 2011.
[3] Y. Fang, F. Wang, and J. Ge, “A task scheduling algorithm based on load balancing in cloud comput-ing,” Web Information Systems and Mining, pp. 271–277, 2010.
[4] S. Parsa and R. Entezari-Maleki, “Rasa: A new task scheduling algorithm in grid environment,” World Applied sciences journal, vol. 7, no. Special issue of Computer & IT, pp. 152–160, 2009.
[5] L. Wang, S. U. Khan, D. Chen, J. Kołodziej, R. Ranjan, C.-Z. Xu, and A. Zomaya, “Energy-aware parallel task scheduling in a cluster,” Future Generation Computer Systems, vol. 29, no. 7, pp. 1661– 1670, 2013.
[6] U. Awada, K. Li, and Y. Shen, “Energy consumption in cloud computing data centers,” International Journal of Cloud Computing and services science, vol. 3, no. 3, p. 145, 2014.
[7] L. Luo, W. Wu, D. Di, F. Zhang, Y. Yan, and Y. Mao, “A resource scheduling algorithm of cloud computing based on energy efficient optimization methods,” in Green Computing Conference (IGCC), 2012 International, pp. 1–6, IEEE, 2012.
[8] S. Javanmardi, M. Shojafar, D. Amendola, N. Cordeschi, H. Liu, and A. Abraham, “Hybrid job scheduling algorithm for cloud computing environment,” in Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014, pp. 43–52, Springer, 2014.
[9] D. Agarwal, S. Jain, et al., “Efficient optimal algorithm of task scheduling in cloud computing envi-ronment,” arXiv preprint arXiv:1404.2076, 2014.
[10] S. J. Patel and U. R. Bhoi, “Improved priority based job scheduling algorithm in cloud comput-ing using iterative method,” in Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on, pp. 199–202, IEEE, 2014.
[11] M.-A. Hassan, I. Kacem, S. Martin, and I. M. Osman, “Genetic algorithms for job scheduling in cloud computing,” Studies in Informatics and Control, vol. 24, no. 4, pp. 387–400, 2015.

[12] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," Egyp-tian informatics journal, vol. 16, no. 3, pp. 275–295, 2015.
[13] K.-M. Cho, P.-W. Tsai, C.-W. Tsai, and C.-S. Yang, "A hybrid meta-heuristic algorithm for vm schedul-ing with load balancing in cloud computing," Neural Computing and Applications, vol. 26, no. 6, pp. 1297–1309, 2015.
[14] A. S. Milani and N. J. Navimipour, "Load balancing mechanisms and techniques in the cloud en-vironments: Systematic literature review and future trends," Journal of Network and Computer Applications, vol. 71, pp. 86–98, 2016.
[15] S. Mittal and A. Katal, "An optimized task scheduling algorithm in cloud computing," in Advanced Computing (IACC), 2016 IEEE 6th International Conference on, pp. 197–202, IEEE, 2016.
[16] S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," Journal of grid computing, vol. 14, no. 2, pp. 217–264, 2016.
[17] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A survey of pso-based scheduling algorithms in cloud computing," Journal of Network and Systems Management, vol. 25, no. 1, pp. 122–158, 2017.
[18] H. Duan, C. Chen, G. Min, and Y. Wu, "Energy-aware scheduling of virtual machines in heteroge-neous cloud computing systems," Future Generation Computer Systems, vol. 74, pp. 142–150, 2017.
[19] S. Lalwani, S. Singhal, R. Kumar, and N. Gupta, "A comprehensive survey: Applications of multi-objective particle swarm optimization (mopso) algorithm," Transactions on Combinatorics, vol. 2, no. 1, pp. 39–101, 2013.
[20] H. Wang, H. Sun, C. Li, S. Rahnamayan, and J.-S. Pan, "Diversity enhanced particle swarm opti-mization with neighborhood search," Information Sciences, vol. 223, pp. 119–135, 2013.
[21] X. Shao, W. Liu, Q. Liu, and C. Zhang, "Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem.," International Journal of Advanced Manufacturing Technology, vol. 67, 2013.
[23] G. Wu, D. Qiu, Y. Yu, W. Pedrycz, M. Ma, and H. Li, "Superior solution guided particle swarm optimization combined with local search techniques," Expert Systems with Applications, vol. 41, no. 16, pp. 7536–7548, 2014. J. Behnamian, "Particle swarm optimization-based algorithm for fuzzy parallel machine schedul-ing," The International Journal of Advanced Manufacturing Technology, vol. 75, no. 5-8, pp. 883–895, 2014.
[24] X. Zuo, G. Zhang, and W. Tan, "Self-adaptive learning pso-based deadline constrained task schedul-ing for hybrid iaas cloud," IEEE Transactions on Automation Science and Engineering, vol. 11, no. 2,
pp. 564–573, 2014.
[25] X. Liang, W. Li, Y. Zhang, and M. Zhou, "An adaptive particle swarm optimization method based on clustering," Soft Computing, vol. 19, no. 2, pp. 431–448, 2015.
[26] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," Mathematical Problems in Engineering, vol. 2015, 2015.
[27] F. Marini and B. Walczak, "Particle swarm optimization (pso). a tutorial," Chemometrics and Intelli-gent Laboratory Systems, vol. 149, pp. 153–165, 2015.
[28] N. Lynn and P. N. Suganthan, "Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation," Swarm and Evolutionary Computation, vol. 24, pp. 11– 24, 2015.
[29] M. Schmitt and R. Wanka, "Particle swarm optimization almost surely finds local optima," Theoreti-cal Computer Science, vol. 561, pp. 57–72, 2015.
[30] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, and A. C. Ammari, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem," Journal of Intelligent Man-ufacturing, pp. 1–13, 2015.
[31] Q. Lin, J. Li, Z. Du, J. Chen, and Z. Ming, "A novel multi-objective particle swarm optimization with multiple search strategies," European Journal of Operational Research, vol. 247, no. 3, pp. 732–744, 2015.
[32] U. C. Allard, G. Dubé, R. Khoury, L. Lamontagne, B. Gosselin, and F. Laviolette, "Time adaptive dual particle swarm optimization," in Evolutionary Computation (CEC), 2017 IEEE Congress on, pp. 2534– 2543, IEEE, 2017.
[33] Z. Jia and X. Fan, "Adaptive particle swarm optimization and svm based novel pattern recognition paradigm," Boletín Técnico, vol. 55, no. 6, pp. 248–254.
[34] X. Zhao and J. Dou, "A hybrid particle swarm optimization approach for design of agri-food supply chain network," in Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference on, pp. 162–167, IEEE, 2011.
[35] M. R. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, and E.-G. Talbi, "A comparative study between dynamic adapted pso and vns for the vehicle routing problem with dynamic requests," Applied Soft Computing, vol. 12, no. 4, pp. 1426–1439, 2012.
[36] G. Zhang, "Hybrid variable neighborhood search for multi objective flexible job shop scheduling problem," in Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th Interna-tional Conference on, pp. 725–729, IEEE, 2012.
[37] G. Nápoles, I. Grau, and R. Bello, "Particle swarm optimization with random sampling in variable neighbourhoods for solving global minimization problems," in International Conference on Swarm Intelligence, pp. 352–353, Springer, 2012.
[38] F. P. Goksal, I. Karaoglan, and F. Altiparmak, "A hybrid discrete particle swarm optimization for ve-hicle routing problem with simultaneous pickup and delivery," Computers & Industrial Engineering, vol. 65, no. 1, pp. 39–53, 2013.
[39] Y. Y. Chen, C. Y. Cheng, L. C. Wang, and T. L. Chen, "A hybrid approach based on the variable neighborhood search and particle swarm optimization for parallel machine scheduling problems a case study for solar cell industry," International Journal of Production Economics, vol. 141, no. 1,
pp. 66–78, 2013.
[40] T. C. Wong and S.-C. Ngan, "A comparison of hybrid genetic algorithm and hybrid particle swarm optimization to minimize makespan for assembly job shop," Applied Soft Computing, vol. 13, no. 3,
pp. 1391–1399, 2013.
[41] A. F. Ali, A. E. Hassanien, V. Snášel, and M. F. Tolba, "A new hybrid particle swarm optimization with variable neighborhood search for solving unconstrained global optimization problems," in Proceed-ings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014, pp. 151–160, Springer, 2014.
[42] P. Ghamisi and J. A. Benediktsson, "Feature selection based on hybridization of genetic algorithm and particle swarm optimization," IEEE Geoscience and Remote Sensing Letters, vol. 12, no. 2,
pp. 309–313, 2015.
[43] X. Wang, Y. Shi, D. Ding, and X. Gu, "Double global optimum genetic algorithm–particle swarm optimization-based welding robot path planning," Engineering Optimization, vol. 48, no. 2, pp. 299– 316, 2016.

[44] Y. Marinakis, A. Migdalas, and A. Sifaleras, "A hybrid particle swarm optimization–variable neigh-borhood search algorithm for constrained shortest path problems," European Journal of Operational Research, vol. 261, no. 3, pp. 819–834, 2017.
[45] S.-M. Chen and C.-Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," Expert Systems with Applications, vol. 38, no. 12, pp. 14439–14450, 2011.
[46] M. S. Kıran, E. Özceylan, M. Gündüz, and T. Paksoy, "A novel hybrid approach based on particle swarm optimization and ant colony algorithm to forecast energy demand of turkey," Energy conver-sion and management, vol. 53, no. 1, pp. 75–83, 2012.
[47] S. Arora and S. Singh, "A conceptual comparison of firefly algorithm, bat algorithm and cuckoo search," in Control Computing Communication & Materials (ICCCCM), 2013 International Conference on, pp. 1–4, IEEE, 2013.
[48] R. Raju, R. Babukarthik, and P. Dhavachelvan, "Hybrid ant colony optimization and cuckoo search algorithm for job scheduling," in Advances in Computing and Information Technology, pp. 491–501, Springer, 2013.
[49] R. Rahmani, R. Yusof, M. Seyedmahmoudian, and S. Mekhilef, "Hybrid technique of ant colony and particle swarm optimization for short term wind energy forecasting," Journal of Wind Engineering and Industrial Aerodynamics, vol. 123, pp. 163–170, 2013.
[50] A. Ghanbari, S. M. Kazemi, F. Mehmanpazir, and M. M. Nakhostin, "A cooperative ant colony optimization-genetic algorithm approach for construction of energy demand forecasting knowledge-based expert systems," Knowledge-Based Systems, vol. 39, pp. 194–206, 2013.
[51] C.-L. Huang, W.-C. Huang, H.-Y. Chang, Y.-C. Yeh, and C.-Y. Tsai, "Hybridization strategies for con-tinuous ant colony optimization and particle swarm optimization applied to data clustering," Applied Soft Computing, vol. 13, no. 9, pp. 3864–3872, 2013.
[52] M. F. Sulaima, M. F. Mohamad, M. H. Jali, W. Bukhari, and M. Baharom, "Comparative study of heuristic algorithm abc and ga considering vpi for network reconfiguration," in Power Engineering and Optimization Conference (PEOCO), 2014 IEEE 8th International, pp. 182–187, IEEE, 2014.
[53] P. Pongchairerks, "Variable neighbourhood search algorithms applied to job-shop scheduling prob-lems," International Journal of Mathematics in Operational Research, vol. 6, no. 6, pp. 752–774, 2014.
[54] C.-Y. Liu, C.-M. Zou, and P. Wu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing," in Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2014 13th International Symposium on, pp. 68–72, IEEE, 2014.
[55] J. Zhao and H. Qiu, "Genetic algorithm and ant colony algorithm based energy-efficient task scheduling," in Information Science and Technology (ICIST), 2013 International Conference on,
[56] 946–950, IEEE, 2013.T.-S. Pan, T.-K. Dao, S.-C. Chu, et al., "Hybrid particle swarm optimization with bat algorithm," in Genetic and evolutionary computing, pp. 37–47, Springer, 2015.
[57] G. Kanagaraj, S. Ponnambalam, and N. Jawahar, "A hybrid cuckoo search and genetic algorithm for reliability–redundancy allocation problems," Computers & Industrial Engineering, vol. 66, no. 4,
pp. 1115–1124, 2013.
[58] S. Nesmachnow, H. Cancela, and E. Alba, "A parallel micro evolutionary algorithm for heteroge-neous computing and grid scheduling," Applied Soft Computing, vol. 12, no. 2, pp. 626–639, 2012.
[59] J. A. Parejo, A. Ruiz-Cortés, S. Lozano, and P. Fernandez, "Metaheuristic optimization frameworks: a survey and benchmarking," Soft Computing, vol. 16, no. 3, pp. 527–561, 2012.
[60] S. Nesmachnow, "Parallel multiobjective evolutionary algorithms for batch scheduling in heteroge-neous computing and grid systems," Computational Optimization and Applications, vol. 55, no. 2,
pp. 515–544, 2013.
[61] I. BoussaïD, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," Information Sci-ences, vol. 237, pp. 82–117, 2013.
[62] V. Kachitvichyanukul and S. Sitthitham, "A two-stage genetic algorithm for multi-objective job shop scheduling problems," Journal of Intelligent Manufacturing, vol. 22, no. 3, pp. 355–365, 2011.
[63] X. Li, L. Gao, and W. Li, "Application of game theory based hybrid algorithm for multi-objective inte-grated process planning and scheduling," Expert Systems with Applications, vol. 39, no. 1, pp. 288– 297, 2012.
[64] N. Shahsavari-Pour and B. Ghasemishabankareh, "A novel hybrid meta-heuristic algorithm for solv-ing multi objective flexible job shop scheduling," Journal of Manufacturing Systems, vol. 32, no. 4,
pp. 771–780, 2013.
[65] J. Wang, B. Gong, H. Liu, S. Li, and J. Yi, "Heterogeneous computing and grid scheduling with parallel biologically inspired hybrid heuristics," Transactions of the Institute of Measurement and Control, vol. 36, no. 6, pp. 805–814, 2014.
[66] F. Ramezani, J. Lu, J. Taheri, and F. K. Hussain, "Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments," World Wide Web, vol. 18, no. 6, pp. 1737– 1757, 2015.
[67] N. Kumar and D. P. Vidyarthi, "A novel hybrid pso–ga meta-heuristic for scheduling of dag with communication on multiprocessor systems," Engineering with Computers, vol. 32, no. 1, pp. 35–47, 2016.
[68] S. Kardani-Moghaddam, F. Khodadadi, R. Entezari-Maleki, and A. Movaghar, "A hybrid genetic algorithm and variable neighborhood search for task scheduling problem in grid environment," Procedia Engineering, vol. 29, pp. 3808–3814, 2012.
[69] S. Nesmachnow, "An overview of metaheuristics: accurate and efficient methods for optimisation," International Journal of Metaheuristics, vol. 3, no. 4, pp. 320–347, 2014.
[70] K. Joshi, K. Jain, and V. Bilolikar, "A vns-ga-based hybrid metaheuristics for resource constrained project scheduling problem," International Journal of Operational Research, vol. 27, no. 3, pp. 437– 449, 2016.
[71] G. Yao, Y. Ding, Y. Jin, and K. Hao, "Endocrine-based coevolutionary multi-swarm for multi-objective workflow scheduling in a cloud system," Soft Computing, vol. 21, no. 15, pp. 4309–4322, 2017.
[72] Y. Wen, H. Xu, and J. Yang, "A heuristic-based hybrid genetic-variable neighborhood search algo-rithm for task scheduling in heterogeneous multiprocessor system," Information Sciences, vol. 181, no. 3, pp. 567–581, 2011.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY