JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.433-450
https://publishoa.com
ISSN: 1309-3452

FLLBHGATS: Efficient Load Balancing and Task Scheduling Algorithm for Real-Time Multiprocessor

NirmalaH

Corresponding author, Research Scholar, Department of Computer Science and Engineering,

RNS Institute of Technology, Bangalore.

GirijammaH A

Professor, Department of Computer Science and Engineering, RNS Institute of Technology,

Bangalore.

Received 2022 April 02**; Revised** 2022 May 20**; Accepted** 2022 June 18**.**

Abstract

Different experiment has been advertised that the processor work load distributing equitably with the processors of a distributed system decidedly enhance framework execution and improves system management. Fuzzy logic has been implemented in numerous areas of industry and science to manage susceptibility. Proposed work with the intent of load balancing has been focused on using fuzzy logic to interpret processor's load and task execution length. This work introduces a new dynamic fuzzy-based load balancing algorithm for homogeneous dispersed frameworks. The proposed techniques use fuzzy logic to manage improper data load i.e., overloaded and under loaded, deciding on load distribution choices and preserve general framework strength. For accurately evaluating the load status of a host, proposed algorithm uses CPU utilization, CPU queue length and distance upon its present load as linguistic inputs while framing fuzzy set. Method proposes Hybrid Genetic Algorithm (HGA) that is blended with stochastic development process in order to designate and schedule real-time tasks with priority requirements. The work randomly generates the tasks using random wheel approach, once the tasks are generated then encoding tasks to chromosome is carried out. Height of each task is obtained through DAG and according to the root node, the height of each taskis updated in the chromosome. Proposed fuzzylogicbased load balancing and hybrid genetic algorithm based task scheduling (FLLBHGATS) algorithm has been evaluated with similar existing methods in order to prove its efficiency. The results prove that FLLBHGATS performs better than other techniques as far as the solution quality.

Keywords: Fuzzy Logic, Genetic Algorithm, Load balancing, Multi-processor, Task Scheduling.

1. INTRODUCTION

Numerous works has shown that workload distribution equally with processorsof a distributed system vastly enhances framework execution and maintainsthe resources utilization. Load balancing in distributed frameworks will becharacterized the way towards reallocating the work among processors withinthe framework to enhance framework execution [1]. Battery powered devicesdepend more on high sustainable processors and are equipped for functioning inreal-time operations (e.g., voice and movie acknowledgment) [2, 3]. The circumstance necessities real-time system recognization with non-real-timeframeworks. Compare to non-real-time frameworks, real-time system should createsensibly perfect outcomes within a cutoff time. Since processors devour tonsof energy in compute systems, a considerable measure of task has put on theenergy minimizing devoured by the processors. DVS is the simplest procedureused to diminish processor energy usage. DVS permits powerfully mount boththe voltage and handling processor's frequency at run time and whatever pointsthe complete handling advance isn't needed [4]. This, makes tasks taking longerperiod to end and subsequently, a couple of tasks may miss their deadline. Inhard real-time multiprocessor frameworks, selecting a correct processor and relating working voltage/frequency is very essential. This factor offers rise in the energy-efficient task scheduling.

Dynamic load balancing methods supervise on the framework workload andrearrange the workload appropriately [5]. An effective load balancing algorithm is generally made out of3 approaches, i.e., information, location and transferapproach. In information approach, the data focuses on load balancing process.Location approach executes a transferred task using remote node. Transfer approach settles on the tasks that are qualified for a move to different nodes forpreparation. Information approach assigns location and transfer methodologiesat every node with the vital data necessary for decision. Information system will be a significant factor for load balancing process. Cost and intricacy ofany powerful load balancing process relies vigorously upon the task of data technique. The execution authority or dynamic load balancing process control cantake 3 distinct structures such as centralized, semi-distributed and distributed.Centralized load balancing process, will have a dedicated solitary node (calleda focal node) gathers the information about the state of the system and makeuses it to load balancing decisions within the organization. In distributed loadbalancing, the load is disseminated and every node in the organization conveysthe same portion of the requirement and executes the same process. Semi-distributed load balancing divides the entire organization into clusters, each with its own set of nodes.Furthermore, each cluster's node control is centralized.Load adjustment in distributed systems is performed in this method by involving the focal nodes of each cluster.For example, task is distributed among each cluster's focal nodes.

Despite the ideal fact that has been implemented on real-time tasks scheduling on distribution and multiprocessor frameworks, there are so far broad exploration effort to progress improved and productive task distribution andscheduling methods down various situations and framework necessities. In thiswork, scheduling of real-time task on proposed multiprocessor frameworks hasan optimization issue exposed to a set of limitations. The goal is to limit theenergy utilization task priority and deadline requirements. To address this issue, system proposes a GA that is hybridized with a stochastic developmentprocess to distribute and to schedule real-time task based on proposed multi-processor frameworks. This methodology incorporates the task allocation toprocessors, task scheduling on all processors and decides the working voltage onwhich the task is being executed into a solitary issue. System additionally being specific hybrid and irritate tasks even as a geography safeguarding algorithmto make the initial population. The working of the proposed technique hasbeen researched through complete reproduction. The efficiency of the proposedwork has been contrasted with various notable meta-heuristics and as a result, the results reveal that proposed method beats other metaheuristics in terms of arrangement quality. The paper is arranged with related work in Section 2 and proposed systemproblem statement in Section 3. Section 4 provides algorithm proposed andevaluation of the proposed work has been presented in 5. In Section 6, conclusionhas been presented.

2. RELATED WORKS

Many works have been proposed in order to balance the load in which, Grosuet al. [6], introduced a game-hypothetical system for retrieving a fair load balancing plan. The principal objective was to infer a fair and ideal distribution plan. They defined the heap adjusting issue in single class work distributedframeworks as a united game among systems and its likewise in proper arrangement. Grosu et al. [7], designed a game-theoretic framework for load balancingindistributed heterogeneous systems. The author proposed non-cooperativeload balancing and presented the Nash equilibrium. Based on the Nash equilibrium, new algorithm is proposed. In this work, planned the heap adjustingissue in heterogeneous appropriated frameworks as a non-agreeable game amongclients. It has low complexity when compared to other techniques and optimumallocation for each user. Nikravan et al. [8], proposed genetic algorithm tosolve process scheduling problem in distributed systems. Algorithm uses heuristic search method to obtain optimal and suboptimal solutions. This solves theNP-complete problem in distributed operating system. Hence, they analyze thealgorithm performance with all possibilities. Computationally expensive andtime consuming are the limitations of this work. Ali M. Alakeel [9], proposed a fuzzy load balancing algorithm to increase system performance by determining when the load balancing process should be started. It helps the overloaded systemto transfer some of its data to the under loaded systems. This performs load balancing process in a right way. It works better, when the network has lessnumber of nodes. But it fails in a large network with thousands of nodes.

Awadalla et al. [10], proposed a modified PSO variant by using two algorithms namely min-min and priority assignment algorithms. These algorithmsminimize the iterations when same problem occurs again and again. And italso focused on energy consumption between full-chip and pre-core DVFS processors. The limitation is that there is no time partitioning technique in thiswork. It leads algorithm to fail in giving best results. HyunJin Kim [11], focused mainly on energy consumption by the processors while processing highlycomputational tasks. Hence, they proposed ant colony optimization techniquefor voltage selection and for tasks scheduling. Here, they used artificial agents toperform desired work. XinXin Mei [12], proposed a work with aim of minimizingthe energy consumption of processors while processing a task. They developedheuristic scheduling algorithm with clusters to compute the frequency or voltageconsumed by each task. This model defines the nonlinear relationship betweentime taken by task to execute and speed of a processor for GPU-acceleratedapplications. This work contains more assumptions while solving the problems.Hence, there is no practical formulation on the accurate results.

Ziranpeng [13], proposed an energy saving strategy for mobile terminals. Itallocates the tasks between mobile terminals under two constraints, this include accomplishing difficult real-time activities and meeting certain energy management needs. Hence, the algorithm worked on dynamic optimization strategy to schedule thetasks. But the limitation of the proposed work is it takes more calculations,which leads to overhead in the system. Alahmad et al. [14], proposed scheme forsolving the problem of distributing the tasks between fixed set of heterogeneousprocessors. Hence, they provide a scheme to allocate tasks to processors based

on speed of processor and computation time. Hence, this work helps to provideQoS and reduces energy consumption of the system. But the limitation of this work is it will not provide the extensive measurements. Hence, it fails toestimate the interface between the tasks which have mutual cache. Gharbietal. [15], proposed a hybrid genetic strategy for real-time scheduling on crucial multiprocessors using low power. They demonstrated that the hybrid genetic technique outperforms the traditional genetic strategy. HGA works wellin balancing the load with less response time and with good exibility. Multiprocessors which deals with dependent and independent tasks are not addressedin this work and also don’t work with data transfer and data managementtasks. Viswanathan et al. [16], proposed RADIS methodologies which efficientlyhandles the large loads. The large loads are divided by the concept of DLT. Inthis work the large loads which are divided into sub loads, and sub loads whichare not dependent, then that tasks are assigned to the nodes. The real time data are used during the simulation. Zhu et al. [17], suggested two planning methodsfor tasks with or without priority limitations in multi-processor frameworks. Theproposed method decreases energy utilization by decreasing speed in recoveryof utilized time by tasks. Tavares et al. [18], proposed Petri-nets technique for real-time task scheduling with voltage scaling deadline. Author's used primary-run time approach rather than scheduling run-time approach to ensure that each one of the tasks has to find its deadlines.

3. PROBLEM STATEMENT

The problem statement of the proposed work is:

1. Estimating the workload of a system and to distribute load equally amongthe systems which are either overloaded or under loaded using load balancing scheme.
2. Scheduling of the tasks to the processor with minimized energy consumption.

4. PROPOSED SYSTEM

In this section, fuzzy logic is used for load balancing to share the load equallyin the distributed network and genetic algorithm for minimizing energy usage in real-time task scheduling for the multiprocessor environment.

4.1. Fuzzy logic for load balancing

4.1.1. The System Model

The proposed framework model has been explored in this section.Model hasN number of systems, where N > 1, autonomous systems that are connectedto an operating range and each system comprises of multiprocessors. Tasks thatarrive at a system will be either from outside the organization or with differentsystems within the organization. Systems are exposed to an identical normalappearance rate of working tasks expected from outside the organization. Every distributed job has been queued and prepared on a First Come First Serve (FCFS) basis at each system.

The following assumptions underpin the proposed system:

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.433-450
https://publishoa.com
ISSN: 1309-3452

- The distributed system's nodes ranges from 1 to N, with Ni being the total nodes in system and each node is labeled with Identification (ID).
- Because each node in system is connected via a broadcast network, the cost of transmitting a message between any two nodes will always be the same.
- The system accepts all processors in any state S_k and its correspondingtasks T_k. Initially the distributed system will be in steady state.
- From proposed design, the load balancing process attempts to review theoverall condition of the system and makes the important restorative moveslikewise in accord to the goals which the algorithm expects.

4.1.2 The Objectives

In Distributed system, when a node is chosen for load balancing, the systemshould perform the following objectives:

1. It has to efficiently access complete data about the load of every systemwithin the framework.
2. To keep a load balancing in the distributed framework this is a requirement for clear distinction d. Estimation of *d*differs over activityof load balancing process and balances it progressively considering thestatus of the framework and the data communication values. The legitimate *d* range could also be resolved after experiments with this processand this cycle must be done efficiently regarding the correspondence timeneeded.
3. To choose the ultimate proper opportunity to dispatch the load balancingmeasure. Keeping the load balance performing consistently may be aweight on the framework, so a proficient method of setting off the loadbalance is being considered. The load balancing is considered if any oneamong the accompanying conditions is fulfilled:

a. At the purpose when a system gets inactive or under loaded.

b. At the purpose when a system gets overloaded.

4. To guarantee that system is providing the load balancing at a necessarytime. It's conceivable that one system can meet both of the circumstances.This is able to make multiple load balances to be dynamic simultaneously.To prevent this, system algorithm guarantees that only one load balanceis dynamic at a time.

4.1.3 The Algorithm Steps

The load balancing system performs the following steps in the fuzzy based load balancing process:

1. Get the system's current load.This is accomplished by sending a status message broadcast to all nodes in the system.
2. After receiving the response messages from each node, the load balancing system assigns a fuzzy value within the range [0,1] to each node in the system, including itself, that addresses the node's load while also relating to the overall load of the distributed system. Task will be accomplished through fuzzy set framing, LOADED = {Low, Medium, High} which addresses system load. From fuzzy logic, the accuracy of evaluating the load status of a host, employ the {CPU utilization, CPU queue length, Distance} upon its present load. These values will be within the range [0,1] and follow resemblance of a system's load to the fuzzy term LOADED, which is addressed through fuzzy set The task of participation value depends on the fuzzy set rules which take if-then-else rule format and membership values assigned to this is based on the triangular function. The system includes a fuzzy inference mechanism that takes into accountof 27 rules in which 13 rules are defined in the Table 1. Where Normal represents the state for which no load balancing is required. Overloaded & Underloaded state requires load balancing. These 27 rules are explained in the following three cases.

- If (CPU queue length = Low && CPU utilization = Medium && Distance = Low) Then “Normal"
- If (CPU queue length = Low && CPU utilization = Low && Distance = Low) Then “Underloaded"
- If (CPU queue length = High && CPU utilization = High && Distance = High) Then “Overloaded"

3. In the defuzzification stage, an accurate output value is extracted from the fuzzy sets. For the defuzzification process, weighted mean method is being considered and is formulated as.

$$Z = \sum\eta_0(\bar{O}) \,/\, \sum\eta_0(\bar{O}) \qquad (1)$$

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.433-450
https://publishoa.com
ISSN: 1309-3452
Where Z is derived output value, $\eta_0(\bar{O})$ strength of output membership function and $(\bar{O})$ is centroid of membership function.

4. Utilizing the outcome of step (3), the load balancing orders every node into 3 different states: Underloaded, Normal, and Overloaded. The Normal node doesn't require any load balancing, whereas Underloaded and Overloaded systems will require load balancing measure.

5. Make a mapping from Overloaded→Underloaded system. Results advise that every overloaded systemshould able to move some of its extra work. Due to this, the load balancer communicates to overloaded system with a message determining the ID of every conceivable underloaded nodes and therefore number of tasks that overloaded nodes contains, is made available to underloaded system. This data is framed as index: (ID1, htasks), (ID2, htasks),…., (IDN, htasks). To perform this process, system receive probability model by utilizing load at every system and registers probability ofsending tasks from an overloaded system *i*to an underloaded system *j*.

Table 1: Fuzzy Rule set

Rule	CPU queue Length	CPU utilization	Distance	State
1	L	L	L	Underloaded
2	L	H	L	Overloaded
3	L	H	M	Overloaded
4	L	L	H	Normal
5	L	H	H	Overloaded
6	M	H	L	Overloaded
7	M	H	M	Overloaded
8	M	L	M	Underloaded
9	M	L	H	Normal
10	H	M	L	Normal
11	H	M	M	Normal
12	H	H	L	Overloaded
13	H	L	H	Underloaded

4.2.Hybrid Genetic Algorithm for Efficient Task Scheduling

GA is being used as an efficient method for tackling optimization problems. The GA successfully generates the benefit of global spaces for searching for better optimal solutions to the problem. GA operators like, the crossoverand mutation can be adjusted in like manner with the end goal with the purpose of making them relevant to the problem. Likewise, formation of initial populationcomprising of possible arrangements generally affects the general exhibition.The hybridization in the proposed method has been effectively utilized to accomplish enhanced quality arrangements. Hybrid method has been obtained bycombining highlights of one another heuristic for acquiring ideal or near idealarrangements. Hybrid approaches have commonly shown good exhibitions contrasted with their particular individual heuristics. Stochastic development isa run test iterative search method handed down to conduct various biologicalcycles. During its execution, the algorithm keeps up and works in an iterativeway to gradually build the feasible solutions.

4.2.1 Task Model

T = {t_1, t_2,…., t_N}real-time tasks to be executed on a multiprocessor system. Each task t_i is characterized as (c_i, d_i), here c_i addresses theworst-case computational scheme (in various cycles) and d_i is t_i's task deadline. Every task's clock cycle number is chosen beforehand to ensure that the tasks are non-preemptive and not interfered during execution.Task can disseminate with other tasks & so have priority links.Directed acyclic graph DAG is used to address jobs with priority constraints (T, E)T stands for task sets, while E stands for directed arcs or edges that express dependencies between tasks. An edge $e_{i,j}\epsilon E$ between task t_i and t_j addresses that task $t_i$finish its executionbefore t_j begins. From each edge, $e_{i,j}\epsilon E$, and v_{ij}that addresses the measure of data communicated from t_i to t_j . Figure 1 shows the task graph.

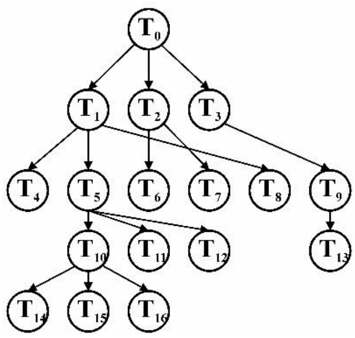


Figure 1: DAG

From task scheduling model, system characterizes $Pre(t_i) = t_j|t_j\epsilon T,e_{i,j}\epsilon E$ as a actual procedures set of task tj and $Succ(t_i) = t_j|t_j\epsilon T,e_{i,j}\epsilon E$ as animmediate predecessors set that task replacement of t_i. If $Pre(t_i)$ is that predecessors task sett$_{i,}$ $t_i$can't begin its execution except if everything of its tasks are done. Besides a $Pre(t_i) = t_j,t_k,t_l...,t_p$may be a predecessors set of tasks t_i. The $e_{jk},e_{kl},...,e_{(p-1)p},e_{pi}E$ &$aPre(t_j) = \emptyset$, at that time there's an directed path from $t_p$to$t_i$and $t_p$doesn't carry any predecessors. In Figure 1, it can be observed that

$Succ(t_6) = \emptyset$& $Succ(t_2) = t_6, t_7$

$Pre(t_3) = t_0$ & $Pre(t_9) = t_3$

$aPre(t_{10}) = t_5, t_1, t_0$ &$aPre(t_7) = t_2, t_0$

The total execution time, τ_i, of tasks $t_i$at frequency f_i is given by:

$$\tau i = \frac{c_i}{f_i}(2)$$

The proposed method has two reserve times: the Newest Start Time (NST) and the Earliest Start Time (EST).If job fails to start at this time, the task's NST should start it at this time.Then it risks missing the deadline.If d_i is task completion time and t_i is deadline time, at that point NST is given by:

$$NST_i= d_i - \tau_i(3)$$

The EST of a task $t_i(EST_i)$ is time beforeprocess begins, and it is formulated as follows:

$$EST_i = \begin{cases} max\{FT_j\}, \\ 0, \quad ifPre(t_i) = \emptyset \end{cases}(4)$$

Task $t_i$never skips its deadline time, if its actual execution time (ST_i) exists in the NST_iandEST_i. That is,

$$EST_i\leq ST_i\leq NST_i \ (5)$$

Finally, the end time of task $t_i$is given by

$$ET_i= ST_i+ \tau_i(6)$$

4.2.2 Energy Model

Consider a proposed multiprocessor framework with M processors $\{p_1,p_2,...p_M\}$. Every processor is efficient for working on various discrete voltage levels. The system expects that $p_k$processor has $l_k$various voltage levels. In working of voltage level can be progressively and quickly adapts to any working levels of voltage, individually of different

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.433-450
https://publishoa.com
ISSN: 1309-3452
processors. If $c_{ef}$is effective switching capacitance, $V_i$is voltage supply, and f_i is working (frequency obtained after run time) on that task $t_i$is executed, so consumption of power utilized during this process is given by [19]:

$power_i = c_{ef} \times V_i^2 \times f_i$(7)

Interface with power and voltage is as follows

$$f_i = \xi \times \frac{(V_i - V_t)^2}{V_i} (8)$$

The ξ circuit dependent, V_t as voltage threshold and $V_t << V_i$. It is critical that the processor frequency be reduced in tandem with the system voltage.Furthermore, while the number of clock cycles tasks t_i are known in advance and fixed, their execution time may vary when the processor frequency changes.

$e_i = power_i \times \tau_i$ (9)

i.e.

$e_i = c_{ef} \times V_i^2 \times c_i$(10)

As per Eq. (10), the energy used during clock cycle is relative to system voltage squared. Accordingly, minor alteration in the working processor voltage can bring huge variation in energy utilization. Subsequently, energy usage is limited by regulating processing voltage.

4.2.3 Objective Function and Constraints

Eq. (11) specifies that the task $t_i$at voltage $v_l$energy used. Entire energy used by every task$E$is summarized as:

$$E = \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{l}^{|v_k|} x_{ikl} \times e_{ikl} (11)$$

where e_{ikl}indicates energy used by task $t_k$done on processor $p_j$at a voltage level v_k, and x_{ikl}be a result variable which is determined as:

$$x_{ikl} = \begin{cases} 1, & if\ task\ t_i\ allocated\ to\ p_k\ at\ voltage\ level\ l \\ 0, & Otherwise \end{cases} (12)$$

Problem of task scheduling touted as a 0–1 decision problem to optimise E within specific constraints. It must keep this to a minimum here.

$\sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{l}^{|v_k|} x_{ikl} \times e_{ikl}$(13)

Dependent to

$ST_i \geq EST for\ each\ i, 1 \leq i \leq N$ (14)

$ET_i \leq d_i foreach\ i, 1 \leq i \leq N$ (15)

$$\sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{l}^{|v_k|} x_{ikl} = 1 \qquad for\ each\ i, 1 \leq i \leq N\ (16)$$

Limitation Eq. (14) determines that a task can’t begin before the communication of the entirety of its predecessor tasks. The subsequently limitation Eq. (15) indicates the real time constraints and at last Eq. (16) each task is circumscribed to precisely single processor for a single level of voltage.

4.3. Proposed Hybrid Genetic Algorithm

With the use of fuzzy logic, the system has following three states asNormal, Overloaded and Under loaded. Before allocating the tasks, the proposedfuzzy load balancer will share the load in the network equally to the system.Once the

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.433-450
https://publishoa.com
ISSN: 1309-3452
load is allocated to thesystem. Load allocation strategy is as shownin Algorithm 4.1.3, then the task scheduling is done by the HGA.

The proposed HGA starts with initialization of all the necessary parametersas shown in the Algorithm 1. After all the initialization, it randomly generatesthe tasks using random wheel approach. Once the tasks are generated, thenencoding of the tasks to chromosome is done. Then, it fetches the height ofeach task from DAG and according to the root, the height of the tasks isupdated in the chromosome. Randomly selects the available processors andassigns it to the tasks. Selects the voltage levels randomly and assigns to theprocessor and encodes it to the chromosomes. After encoding the chromosome,check whether the chromosome is feasible, if it is feasible then add the chromosome tothe population. In the Algorithm 3, selects the two chromosomes randomly fromthe population, apply the crossover operation and perform the feasibility test foreach child in the chromosome. The children who failed with the feasibility testare discarded. This iteration is applied until the fitness evaluation is satisfied.Stochastic development has been applied to each child who has passed inthe feasibility test with a p_h. The mutation is applied until the improvement isobtained, if no improvement is observed then the iteration will be stopped.

4.3.1 Chromosome Encoding and Generating the Initial Population

In GA, a chromosome addresses a possible state of scheduling. Each chromosomeis represented with the set of tuples (T,P,V,H) where T denotes the task, Pdenotes the processors, V denotes the voltage and H denotes the height ofthe task. This chromosome shows the task assigned to the processors withvarious voltage and the height of the tasks as demonstrated in Figure 2. A GAchromosomeperchance considered as two structural exhibithaving four layers and Nsegments wherein, N is quantity of assignments. Below figure has fivetasks, assigned to the three processors. These values are added according tothe Figure 1.

Task	0	2	3	4	14
Processor	3	2	1	2	3
Voltage level	2	3	1	2	1
Height	1	2	2	3	5

Figure 2: Chromosome encoding

Algorithm 1	The GA initial solution

Procedure
n=0
while (n<= popSize) do
segNo = 0, pos = 0 & T = TaskSet
while (T ≠ ∅) **do**
segNo++ *Segment number*
t_count = 0 *Number of tasks in segment*
segTasks = ∅ *Current segment task*
for$t_i \varepsilon T$**do**
if (aPre(t_i) = ∅) **then**
segTasks = segTasks U t_i
t_count++
end if
if (p=0) **then**
Initialize start and end positions of the Segment
end if
while (segTasks≠ ∅)**do**
t = random generation of tasks (W)
Random task selection

```
Add t to the chromosome
Assign k to processor t
v = random (1 · · · lk)
Select a voltage level randomly
Assign v to t and p
Assign height to each task according to their DAG
 Adjust the height of the tasks
T = T – t
segTasks = segTasks – t
pos++
end while
end for
end while
if (isFeasible(chromosome)) then
 Add chromosome to population
 n++
end if
end while
end procedure
```

4.3.2 Adjusting the Height

Once the task is ready to execute and if it is having higher priority, tasks are subsequently put to priority queue based on their priority. Theadjust height function is applied to the tasks, which will arrange the tasks inall the feasible ways. By this adjustment, the height of the tasks is updatedfrequently and these updates are dependent on the selected tasks. The localheight concept is used in the adjustment, because the height of the tasks changesover certain period. For example: If T2 is root (first task) then the height ofT2 will be updated to 0 and the dependent tasks of the T2 will be updatedaccordingly.

T_2	T_3	T_4	T_5	T_6	T_7
2	4	1	4	1	0
1	2	1	3	2	3
0	1	1	2	3	3

Figure 3: Height Adjustment

In the Figure 3 the tasks are updated according to the height. The scheduling is done according to the priority, but allocation of the processors will be varied. Figure 4 depicts the initial population generated by the method.

Algorithm2: Feasibility of a solution check

```
Procedure
      function isFeasible (chromosome X)
for (i = 1; I <= N; i++) do
            Calculate ET_i for  task t_i in X
            if (ET_i >= D_i) then
return false
Endif
return true
end for
```

end function
end procedure

Algorithm 3: Adaptive selection using GA for crossover and mutation operation

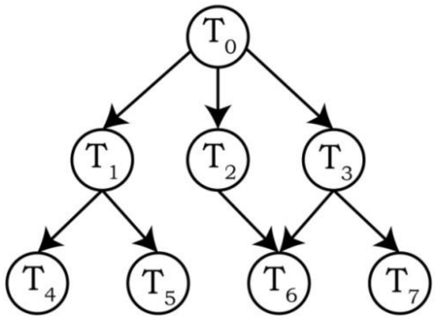
Procedure
Input: Initial population generated by the GA
Output: Fitness evaluation of initial population
while (not termination condition) **do**
Selecting parents from the lot
Perform the Crossover operation
Perform the feasibility test
for (each child X_i) **do**
if (isFeasible (X_i)! = Feasible) **then**
discard $X_i$from the population
end if
for (each child X_i) **do**
if (random () $\leq p_h$) **then**
R=Max generation
ρ=0
cost = cost (X_i)
do {
Select Mutation M_t randomly
X_m= Mutation (X_i, M_t)
if (cost (X_m) ≤ cost &isFeasible (X_m)) **then**
$X_i = X_m$
cost = cost (X_i)
ρ = ρ – R
end if
else ρ++
} **while** (ρ ≤ R)
end if
end for
Fitness evaluation of the children
Replace present population for next generation
end for
end while
end procedure

4.3.3 Crossover Operator

In GA, generation are composed through choosing a couple of chromosomes obtained through current populace with Roulette Wheel Approach (RWA). In proposed algorithm two-pointcrossover is used. The selected chromosomes are applied with crossover to produce the new chromosome. The new chromosome will have the best fitness value than its parent. Figure 5 depicts the first crossover applied to the first two parents to exchange some of the parts between them to produce two different children's chromosomes.

For reproduction of a legitimate chromosome, the requirements listed below must be met:

1. Task heights should vary near the crossover points.
2. Task should all be the same height right before the crossover points.



T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7
6	2	1	5	3	5	1	3
1	3	2	3	1	2	2	3
1	2	2	2	3	3	3	3

T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7
4	6	2	4	1	4	1	0
3	3	1	2	1	3	2	3
1	2	2	2	3	3	3	3

Figure 4: Initial population of chromosomes generated by the method.

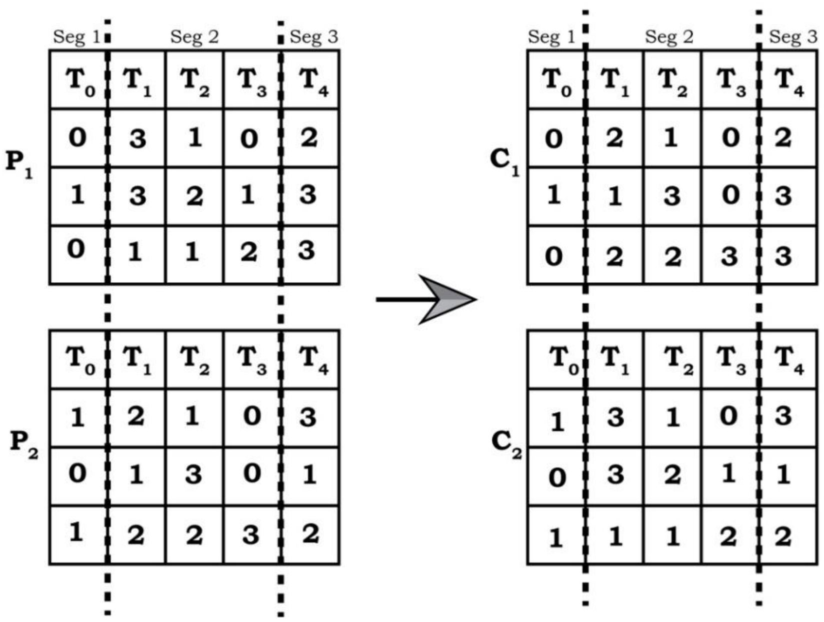


Figure 5: Crossover on the second segment

4.3.4 Mutation Operator

In proposed HGA, the mutation process is supplanted with the mutation which has been generated from stochastic evolution algorithm. After first crossover, a chromosomes subset is chosen for mutation depends on the $p_h$probability. The mutation procedure is carried out as follows for each designated chromosome. The performance of mutation on a randomly chosen quality utilizes a moving compound. Controlled moving compound has been carried out for every type. Compound move with size 2 is accomplished for mutation type 1, with the voltage and processor changing at random. Type 2, 4 &5 single move is done. For type 3, a compound moves of 4 is done, changing the process-voltage pair with another process-voltage pair. The expense of the subsequent arrangement is then assessed. Assuming a decrease in cost is noticed, extra iteration is remunerated to proceed with the mutation. On the off chance, every iteration brings about an improvement and the calculation continue to add additional emphases. Mutation is done until each one of those granted emphases are finished. Nonetheless, in the event if no improvement is noticed then iterations are stopped. Hence, the proposed hereditary calculation carries out the primary elements of moves and the stochastic evolution algorithm's remuneration.Five major distinct sorts of mutation been utilized as follows in the work:

- *Mutation 1:* Here, an arbitrarily chosen gene mutated over exchanging processor & its related voltage level allocated to task.
- *Mutation 2:* In type 2, two randomly selected chromosomes that fall in the similar height, then those genes are swapped.
- *Mutation 3:* In type 3, two genes are exchanged from two randomly selected chromosomes that fall in the same section.
- *Mutation 4:* In type 4, a gene is randomly chosen and exchanged voltage level with the randomly selected voltage level.
- *Mutation 5:* In type 5, a gene is picked at random and mutated by substituting a processor with a number determined at random.

5. PERFORMANCE EVALUATION RESULTS

Performance of proposed algorithm FLLBHGATS and its comparison with the other techniques in the same problem domain is evaluated in this section. The work has been carried out using Python code. For fuzzy load balancing, fuzzywuzzy 0.18.0 library has been considered and implemented the fuzzy algorithm. Four nodes in the network are created and workloads are given as input to fuzzy algorithm so that, the workloads in the network should be shared equally. After the load balancing is done, then the output of the loads are added to the SimSo simulator [20] installed on Windows 10 system with 8 GB RAM. Experiments are performed on the datasets and also section examines experimental considerations and simulation details of FLLBHGATS.

The inputs for algorithm are DAGs, considering differing sizes, tasks deadline time, task height, workload, processors number, operational density and different levelsofvoltage. The studies in this paper use both synthetic and real-time benchmark data.The TGFF tool [21] is consideredtocreate DAGs of various extents using synthetic data. DAGs total tasks considered ranges from 10 to 500. The workloads for the processing tasks, like those used in [22], come from [10, 4500]. On various processors, the optimal execution time for jobs is established by separating the responsibility from the execution speed of the processor. For this, the method proposed by [23] to assign the deadline times to each task, realtime data used from [24], [25] are being used.

5.1 Fuzzy Logic Load Balancing

These two methods [16] and [26] were used to test distributed systems fuzzy load balancing algorithm. Similar simulation setups used in the FLBTS are used and evaluated the proposed fuzzy algorithm.

The proposed fuzzy algorithm when compared for Load vs scheduling delay shown in Figure 6 outperforms than other algorithms. When load is less, it takes more time and when load increases, the delay of scheduling also reduces. Load vs data loss shown in Figure 7 when compared with other algorithms the QBS shows data loss at 3Mb, but FLBTS and the proposed work starts the data loss at 4Mb and proposed work has less data loss compared to FLBTS. Figure 8 depicts the Load vs Throughput when compared with QBS and FLBTS, proposed fuzzyshows the better performance. Load vs success ratio depicted in Figure 9 shows proposed work outperforms when compared to QBS and FLBTS.

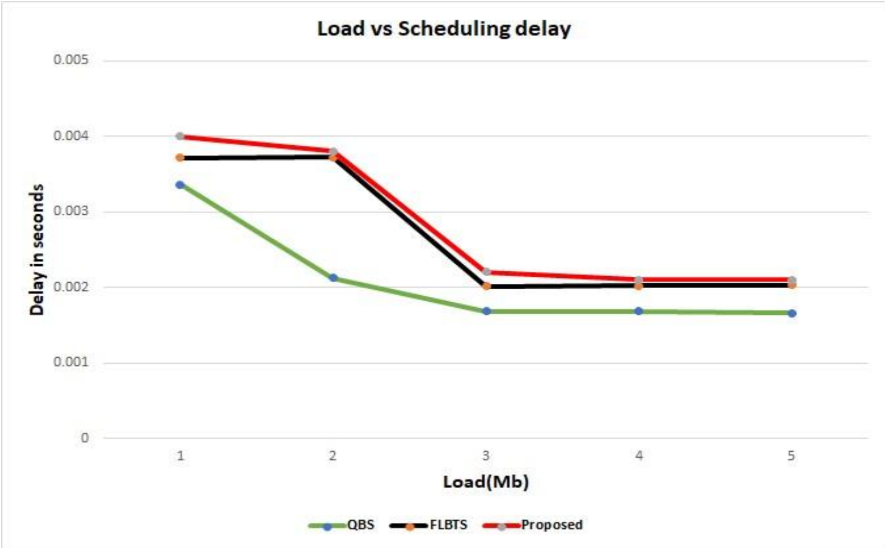


Figure 6: Load vs. scheduling delay

Figure 7: Load vs. data loss

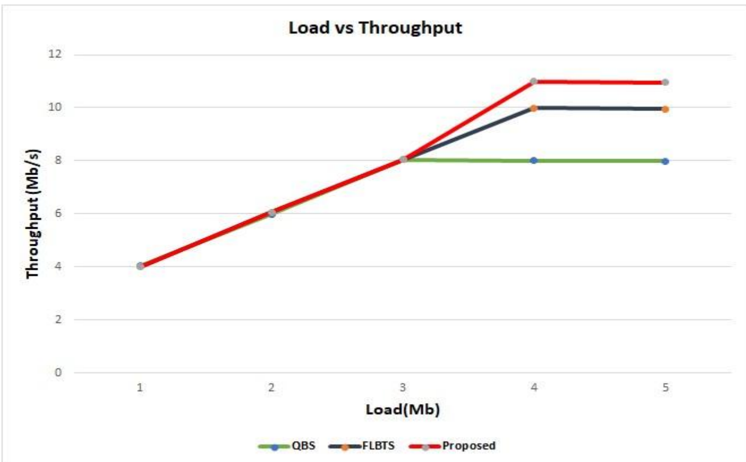


Figure 8: Load vs. throughput

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.433-450
https://publishoa.com
ISSN: 1309-3452

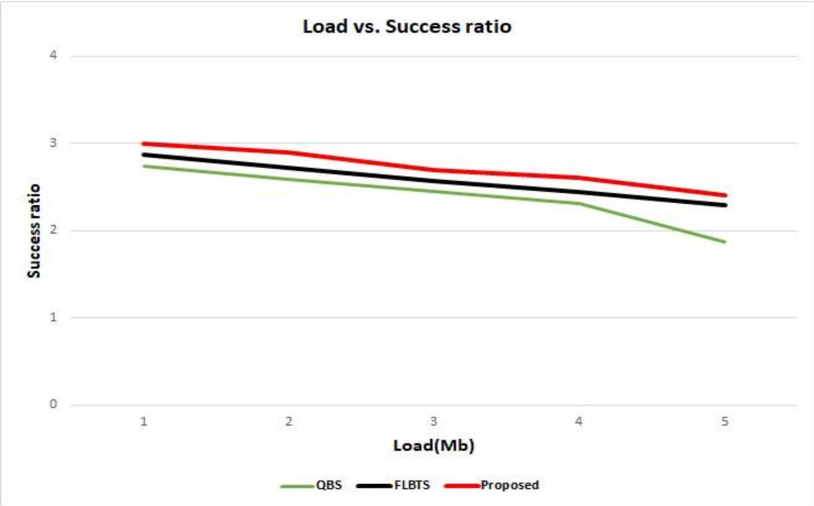


Figure 9: Load vs. success ratio

5.2 HGA Task Scheduling

The proposed hybrid GA (HGA) has been examined with HQIGA & HPSO. For HQIGA, the variation proposed by Konar et al. [27] is being adapted and for HPSO, the work proposed by P. Visalakshi et al. [28] is considered. For reasonable correlations, the population/GA colony size, HPSO, and HQIGA method is taken to be equivalent to the HGA population size that has created the best outcomes for each experiment. All these methods have been simulated for a similar amount of time. Different boundaries utilized for the HPSO and HQIGA method were resolved after experiment and the most appropriate parameter setups are shown in Table2. A similar GA initial population is being used for all the trials in experiment cases and 100 independent runs is performed observing the standard execution for measurably analysing the work of iterative heuristics.

Table 2: Parameter list

Algorithm	Considered Parameter
HGA	System pop size: 40, 80, 100 Rate of Crossover: 0.85 Rate of Mutation: 0.05& 0.1 p_h: 0.1, 0.2, 0.3 Rewarded iterations Φ: 3 & 7
HQIGA	C1=C2=1.67 w=0.52 Vmax=7
HPSO	α = 2 β = 2 P = 0.25 ¥ = 0.3

6 parameters were investigated to determine the ideal parameter setup for the HGA.Initialsize of population, rate of crossover, type of mutation and rate, chromosome select probability ph, and generations of reward are the factors.Table 2 shows the parameter values that were used in the simulation alongside the parameters utilized for different methods. Various arrangement of these parameter values results into an aggregate of 412 combinations. Because of the computational cost engaged with performing tests for all the experiments with the 412 bounded combinations, experiments comprising of 18 and 27 o find the ideal parameter setting for HGA, DAGs are evaluated with all possible combinations. Thus, following combinations delivered the best outcomes: population size = 75, crossover rate = 0.85,

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.433-450
https://publishoa.com
ISSN: 1309-3452
mutation rate = 0.05, p_h= 0.2, and iterations from reward= 7. Earlier mentioned combinations are being used for obtaining experimental results with different DAG's.

Table 3: Comparison of cost for HGA, HQIGA & HPSO

Type of Data	Tasks in No.	HGA	HQIGA	HPSO
Synthetic data	10	41 $\pm$2.00	41 $\pm$2.00	41 $\pm$2.32
	20	66 $\pm$3.49	70 $\pm$2.50	70 $\pm$3.00
	30	80 $\pm$8.59	86 $\pm$0.19	88 $\pm$8.32
	50	110 $\pm$0.23	118 $\pm$2.36	119$\pm$1.89
	70	160 $\pm$2.33	169 $\pm$3.87	170 $\pm$8.96
	100	195 $\pm$3.25	205 $\pm$1.63	209 $\pm$5.56
	200	256 $\pm$1.85	268 $\pm$3.02	275 $\pm$8.53
	400	399 $\pm$9.05	423 $\pm$5.77	424 $\pm$5.02
	500	451 $\pm$4.30	485 $\pm$2.65	493 $\pm$9.53
Real data	45	86 $\pm$7.24	98 $\pm$2.23	102 $\pm$5.69
	100	148 $\pm$3.83	167 $\pm$2.15	173 $\pm$9.57
	400	298 $\pm$6.63	325 $\pm$8.43	330 $\pm$5.56

Table 4: Improved percentage of HGA compared to HQIGA, & HPSO

Data Type	Tasks	HGA vs. HQIGA	HGA vs. PSO
Synthetic Data	10	0.2	2.0
	20	3.901	3.951
	30	13.6	7.73
	50	10.13	10.66
	70	14.54	20.33
	100	14.38	22.31
	200	15.17	11.68
	400	10.72	20.97
	500	28.05	29.23
Real data	45	6.99	14.45
	100	17.7	30.74
	400	28.8	30.93

Table3 displays the processing cost (averaged more than 30 runs) at each employing both synthetic & real-world benchmark data. Outcome of experiment with 18 tasks is shown in synthetic data from the three methods, created consequences of practically a similar quality. Later, for all other remaining cases, HGA delivers the best outcome that is appeared in Table 4 regarding improvement rate. The improvement rate by HGA was in the scope of 0.2% to more than 29.23%. One special case is that the experiment with 18 tasks improvement is appeared by HGA. Statistical testing is being exercised to all the outcomes that are demonstrated practically and all enhancements by HGA were measurably high in rate. In view of the outcomes, it is obvious that HGA has outperformed than other methods. The end is additionally upheld for the pattern search of HGA for two test models that are experimented for 50 to 100 tasks. Figure 10 and 11 shows an average pattern search for HGA in comparison with different methods. It is clear from the two figures that HGA has an option to join to the preferred quality arrangements solutions over different methods. Besides, the quality of the last arrangement got by HGA is superior compared to those which has been obtained by HPSO and HQIGA. These pattern shows that the solid search ability of HGA has empowered it to create better outcomes. With respect to the real benchmark data, similar patterns were seen.Tables 3 and 4 show that HGA produced superior outcome than similar methods investigated, where rate increase achieved by HGA were in scope of 6.99% to 30.93%, & all rate improvement are measurably better.

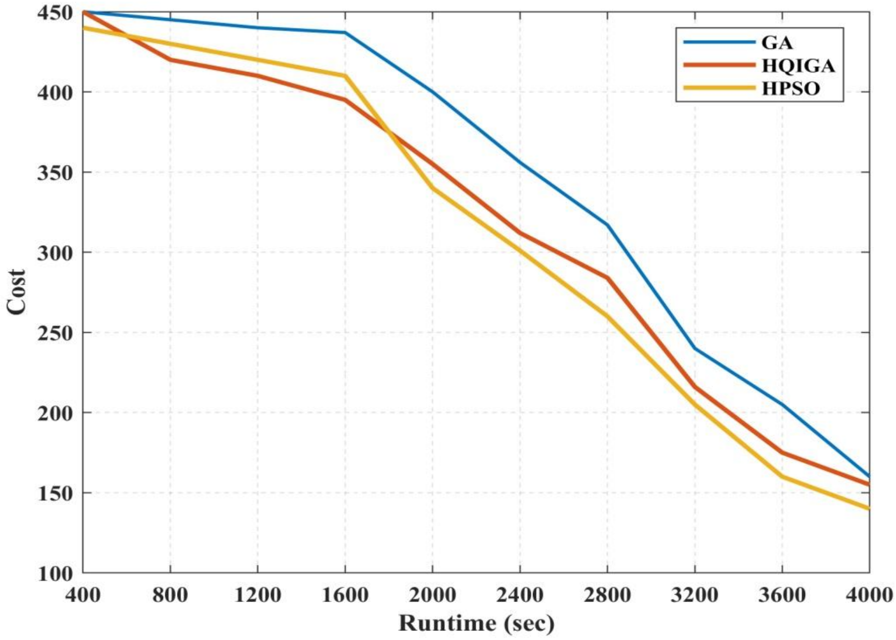


Figure 10: 50 tasks runtime compared with cost.

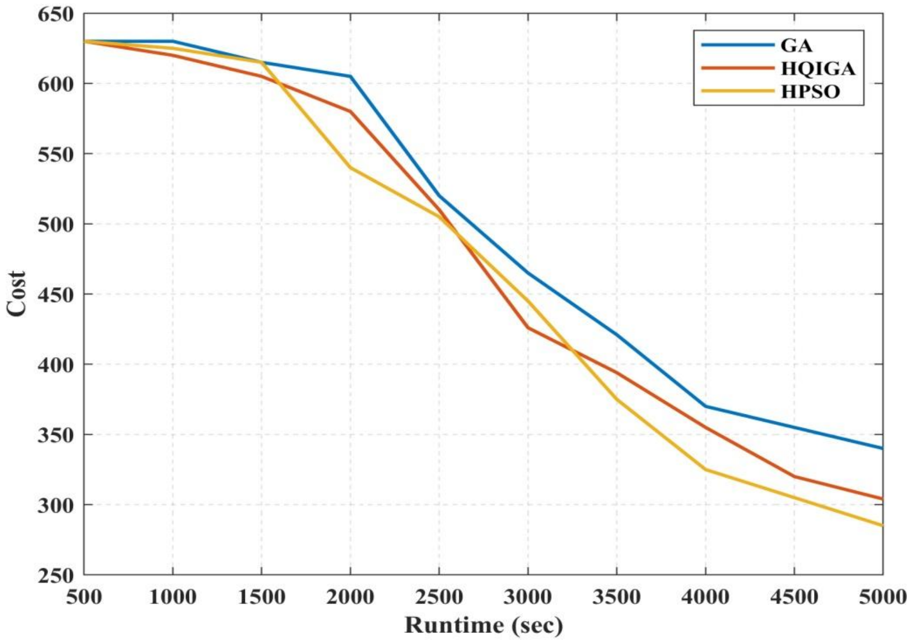


Figure 11: 100 tasks runtime compared with cost.

The total number of irrational resultsHGA algorithm generated while traversing the search space is calculated to provide more insight into the algorithm's searching capacity.Table 5 shows that the % of irrational motions varies among 2% and 12% for synthetic data and between 8% and 10.5% for real-time data.

Table 5: Invalid solutions in percentage

Synthetic Data										Real Data		
No. Of Tasks	10	20	30	50	70	100	200	400	500	45	100	400
Solutions in %	3%	2%	5%	6%	9%	10%	7%	9%	12%	8%	9%	10.5%

6. CONCLUSION

Efficient distribution of workload equally among processors in a distributed system vastly enhances the framework execution time and maintains efficient resources utilization. Load balancing in distributed frameworks will be characterized in way towards reallocating the work among processors which are overloaded and under loaded within the framework to enhance framework execution time. To achieve this fuzzy logic is being used in order to evaluating the load status of a host and to establish load balancing among multiprocessors.Hybrid GA is being used in order to establish efficient scheduling of the task among load balanced processors. The extensive evaluation of proposed FLLBHGATS algorithm proves efficiency of algorithm in par with similar existing algorithms. The proposed method can be further enhanced by considering various intelligent swarm algorithms during task scheduling.

REFERENCES

[1] Ishfaq Ahmad and Arif Ghafoor. Semi-distributed load balancing for massively parallel multicomputer systems. 1991.

[2] KrithiRamamritham, John A. Stankovic, and Wei Zhao. Distributed scheduling of tasks with deadlines and resource requirements. *IEEE Transactions on Computers*, 38(8):1110–1123, 1989.

[3] John A Stankovic, KrithivasanRamamritham, and Shengchang Cheng. Evaluation of a flexible task scheduling algorithm for distributed hard realtime systems. *IEEE Transactions on computers*, 100(12):1130–1143, 1985.

[4] Derek L Eager, Edward D Lazowska, and John Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE transactions on software engineering*, (5):662–675, 1986.

[5] David J Evans and WUN Butt. Dynamic load balancing using task-transfer probabilities. *Parallel Computing*, 19(8):897–916, 1993.

[6] Daniel Grosu, Anthony T Chronopoulos, and Ming-Ying Leung. Load balancing in distributed systems: An approach using cooperative games. In *Proceedings 16th International Parallel and Distributed Processing Symposium*, pages 10–pp. IEEE, 2002.

[7] Daniel Grosu and Anthony T Chronopoulos. Noncooperative load balancing in distributed systems. *Journal of parallel and distributed computing*, 65(9):1022–1034, 2005.

[8] M Nikravan and MH Kashani. A genetic algorithm for process scheduling in distributed operating systems considering load balancing. In *Proc. of 21st European Conference on Modelling and Simulation, ECMS*. Citeseer, 2007.

[9] Ali M Alakeel. Application of fuzzy logic in load balancing of homogenous distributed systems. *International Journal of Computer Science and Security*, 10(3):95–106, 2016.

[10] Medhat Awadalla and Abdullah Elewi. Enhanced pso approach for real time systems scheduling. *International Journal of Computer Theory and Engineering*, 8(4):285, 2016.

[11] HyunJin Kim and Sungho Kang. Communication-aware task scheduling and voltage selection for total energy minimization in a multiprocessor system using ant colony optimization. *Information Sciences*, 181(18):3995–4008, 2011.

[12] Xinxin Mei, Qiang Wang, Xiaowen Chu, Hai Liu, Yiu-Wing Leung, and Zongpeng Li. Energy-aware task scheduling with deadline constraint in dvfs-enabled heterogeneous clusters. *arXiv preprint arXiv:2104.00486*, 2021.

[13] Ziran Peng and Guojun Wang. An optimal energy-saving real-time taskscheduling algorithm for mobile terminals. *International Journal of Distributed Sensor Networks*, 13(5):1550147717707891, 2017.

[14] Bader N Alahmad and Sathish Gopalakrishnan. Energy efficient task partitioning and real-time scheduling on heterogeneous multiprocessor platforms with qos requirements. *Sustainable Computing: Informatics and Systems*, 1(4):314–328, 2011.

[15] Ibrahim Gharbi, Hamza Gharsellaoui, and SadokBouamama. A new hybrid genetic algorithm-based approach for critical multiprocessor realtime scheduling with low power optimization. *Procedia Computer Science*, 159:1547–1557, 2019.

[16] Sivakumar Viswanathan, Bharadwaj Veeravalli, and Thomas G Robertazzi. Resource-aware distributed scheduling strategies for large-scale computational cluster/grid systems. *IEEE transactions on parallel and distributed systems*, 18(10):1450–1461, 2007.

[17] Dakai Zhu, Rami Melhem, and Bruce R Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. *IEEE transactions on parallel and distributed systems*, 14(7):686–700, 2003.

[18] Eduardo Tavares, Paulo Maciel, Bruno Silva, and Meuse N Oliveira Jr. Hard real-time tasks' scheduling considering voltage scaling, precedence and exclusion relations. *Information Processing Letters*, 108(2):50–59, 2008.

[19] Behnam Malakooti, Shaya Sheikh, Camelia Al-Najjar, and Hyun Kim. Multi-objective energy aware multiprocessor scheduling using bat intelligence. *Journal of Intelligent Manufacturing*, 24(4):805–819, 2013.

[20] Maxime Ch´eramy, Pierre-Emmanuel Hladik, and Anne-Marie D´eplanche. Simso: A simulation tool to evaluate real-time multiprocessor scheduling algorithms. In *Proc. of the 5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, WATERS, 2014.

[21] Robert P Dick, David L Rhodes, and Wayne Wolf. Tgff: task graphs for free. In *Proceedings of the Sixth International Workshop on Hardware/Software Codesign.(CODES/CASHE'98)*, pages 97–101. IEEE, 1998.

[22] Guy Martin Tchamgoue, Kyong Hoon Kim, and Yong-Kee Jun. Poweraware scheduling of compositional real-time frameworks. *Journal of Systems and Software*, 102:58–71, 2015.

[23] Patricia Balbastre, Ismael Ripoll, and Alfons Crespo. Minimum deadline calculation for periodic real-time tasks in dynamic priority systems. *IEEE Transactions on computers*, 57(1):96–109, 2007.

[24] Weizhe Zhang, Enci Bai, Hui He, and Albert MK Cheng. Solving energyaware real-time tasks scheduling problem with shuffled frog leaping algorithm on heterogeneous platforms. *Sensors*, 15(6):13778–13804, 2015.

[25] Robert Dick. Embedded system synthesis benchmarks suite (e3s), 2010. *URL: http://ziyang. eecs. umich. edu/˜ dickrp/e3sdd*, 2009.

[26] Abhijit A Rajguru and Sulabha S Apte. Performance improvement of distributed system through load balancing and task scheduling using fuzzy logic. *International Journal of Engineering Systems Modelling and Simulation*, 11(1):35–42, 2019.

[27] DebanjanKonar, Siddhartha Bhattacharyya, Kalpana Sharma, Sital Sharma, and Sri Raj Pradhan. An improved hybrid quantum-inspired genetic algorithm (hqiga) for scheduling of real-time task in multiprocessor system. *Applied Soft Computing*, 53:296–307, 2018.

[28] P Visalakshi and SN Sivanandam. Dynamic task scheduling with load balancing using hybrid particle swarm optimization. *Int. J. Open Problems Compt. Math*, 2(3):475–488, 2018.