Journal of Theoretical and Applied Information Technology
15th March 2024. Vol.102. No 5

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MULTI-STRATEGY BASED FUZZY ENHANCED MONARCH BUTTERFLY OPTIMIZATION ALGORITHM (MS-FEMB0A) FOR TASK SCHEDULING AND LOAD BALANCING IN CLOUD ENVIRONMENTS

[1]DR.SARAVANAN.M.S, [2]MADHAVI KARUMUDI,

1Professor, Department of Computer Science and Engineering,
Saveetha School of Engineering,
Saveetha Institute of Medical and Technical Sciences, Chennai.

2Research Scholar, Saveetha School of Engineering
Assistant Professor, St.Peter's Engineering College
Email: madhavi.karumudi@gmail.com
Email: saravananms@saveetha.com / saranenadu@gmail.com

ABSTRACT

With the success of cloud computing, enterprises started reducing investments on computing resources by using cloud resources. Outsourced tasks of those enterprises are to be processed in cloud without violating Service Level Agreements (SLAs). Therefore, efficient scheduling of tasks in cloud, in presence of millions of users and diversified workloads across the globe, is NP-hard problem. To solve this problem many bio-inspired optimization algorithms came into existence. One such algorithm is known as Monarch Butterfly Optimization (MBO) which suffers from insufficient exploration power and lack of balance between exploration and exploitation. Moreover, most of the algorithms focused either on scheduling or load balancing though they are related. To address all these problems, in this paper, we proposed an algorithm known as Multi-Strategy based Fuzzy Enhanced Monarch Butterfly Optimization Algorithm (MS-FEMB0A) for efficient task scheduling in cloud leading to load balancing. MS-FEMB0A considers multiple objectives and also multiple strategies such as fuzzy, greedy and self-adaptive towards improving search capability and convergence speed. Simulation study made using CloudSim framework showed that MS-FEMB0A outperforms existing algorithms such as PSO and MOPSO.

Keywords – *Cloud Computing, Load Balancing, Task Scheduling, CloudSim, Multi-Strategy Scheduling*

1. INTRODUCTION

Task scheduling in cloud computing is found NP-hard problem due to very dynamic nature of cloud in terms of workloads, heterogeneity of resources and SLAs to be satisfied. Existing techniques used for task scheduling focused on efficient scheduling of tasks but often the load balancing factor is ignored [1]. Therefore, it is very important to consider both to ensure improved performance of cloud infrastructure that lead to consumer satisfaction and growth of service provider due to proper equilibrium. Traditional scheduling approaches like round robin cannot produce optimal results. Moreover, heuristics used traditionally also could not render efficient services. It was found from the literature that evolutionary algorithms could perform better optimization of scheduling in cloud.

Existing methods found in literature contributed towards better scheduling procedures. Evolutionary algorithms studied in [2], [4], [6], to mention few, have potential to deal with different runtime conditions and ensure optimization of scheduling in cloud. At the same time, it was observed that learning based phenomena were widely used in the existing systems. They include swarm intelligence with deep learning [3], online learning based approach [5], deep learning and genetic algorithm [26], hyperparameter optimization based machine learning [39] and oppositional-mutual approach based learning strategy [40]. In [41], MBO is the bio-inspired approach that could improve state of the art in cloud scheduling. However, MBO has specific limitations as discussed in Section 3. In this paper, we enhanced MBO to overcome its drawbacks towards better performance in scheduling and load balancing.

Our contributions in this paper are as follows. We proposed an algorithm known as Multi-Strategy based Fuzzy Enhanced Monarch Butterfly Optimization Algorithm (MS-FEMB0A) for efficient task scheduling in cloud leading to load balancing. MS-FEMB0A considers multiple objectives and also multiple strategies such as fuzzy, greedy and self-adaptive towards improving search capability and convergence speed. The design goals of the proposed algorithm are as follows. First, to enhance MBO functionality to leverage task scheduling performance. Second, to improve balance in exploration and exploitation through self-adaptive strategy. Third, application of greedy strategy towards faster convergence. Fourth, the usage of fuzzy dominance approach towards better solution selection from a set of solutions. Simulation study made using CloudSim framework showed that MS-FEMB0A outperforms existing algorithms such as PSO and MOPSO. The remainder of the paper is structured as follows. Section 2 reviews prior works on scheduling and load balancing in cloud. Section 3 focuses on the preliminary details required by the proposed system. Section 4 presents our method for leveraging state of the art on task scheduling and load balancing. Section 5 presents results of experiments. Section 6 concludes our work.

2. RELATED WORK

This section reviews literature on existing works related to scheduling and load balancing in cloud. Many researchers focused on analysing evolutionary methods that are used in the existing literature for scheduling in cloud [1], [3], [4], [22], [23], [25]. From their study, it was understood that evolutionary methods with swarm intelligence could improve task scheduling performance when compared with traditional approaches. Transfer parameter estimation is a novel approach along with evolutionary methods explored in [2]. With evolutionary algorithms, it was found in the literature, there are contributions that focused on multiple objectives [5], [13], [15], [35], [37]. The essence of these research contributions is to consider different objectives like makespan, balance factor and convergence. They could improve scheduling performance due to their evolutionary nature.

In the existing research, machine learning is also used widely to deal with scheduling in cloud [6], [8], [9], [20]. Evolutionary method was explored in [7] and [10] where intelligence is gained through machine learning. Evolutionary models were found to solve many problems including security issues as discussed in [11] and [30]. Genetic algorithm based approach for workflow scheduling is discussed in [12]. Evolutionary methods for modelling and solving problems is given important in [14], [17] while data-driven approach towards evolutionary optimization is explored in [16]. Deep learning models were also found to be useful for solving problems in real world besides scheduling issues in cloud [8], [9] [17], [18] and [26]. An ensemble of evolutionary algorithm with offline data-driven approach was studied in [19]. Evolutionary approach and its usage along with machine learning towards intelligent systems is the main focus on [21] and [24]. Autoencoding and evolutionary approaches were combined in [27] for automatically understanding the environment and perform multi-tasking.

Along with evolutionary approach a memetic differential technique is exploited with parallel machine scheduling in [28]. Extreme learning machine and learning based approaches are explored in [29] while Random Forest (RF) based machine learning along with evolutionary information is the main focus in [31]. Explainable AI is another important finding in the literature that is used to exploit evolutionary fuzzy systems [32]. Neuro-fuzzy and population-based swarm intelligence were used in [33]. Other significant contributions found in the literature include optimal feature selection [34], gravitational search approach [36], artificial neural network [38], hyperparameter optimization [39] and improved differential evolution algorithm [40]. In [47] and [48] PSO and MOPOSO methods are explored for task scheduling. These re optimization methods based on PSO and the proposed MBO based method is compared with them. From the review of literature, it is understood that enhancing bio-inspired methods with novel strategies has potential to improve scheduling and load balancing performance in cloud.

3. ENHANCED MONARCH BUTTERFLY OPTIMIZATION ALGORITHM

The proposed algorithm in this paper is an extension to an existing algorithm known as Monarch Butterfly Optimization (MBO) explored in [41]. Monarch Butterfly (MB) is a spices in North America. MB has characteristics that differ from others in its behaviour with respect to long

distance migration. MB flies every year from the USA→Canada→Mexico. Its migration behaviour influences the creation of another bio-inspired optimization algorithm known as MBO. Like every swarm intelligence method, exploration and exploitation are two important methods that guide the optimization process. These two methods are performed by MBO in the form of solutions migration and solutions adjusting. With respect to solutions migration, given two sub populations denoted as N_{sp1} and N_{sp2}, their initialization is done as expressed in Eq. 1 and Eq. 2.

$$N_{sp1} = ceil(p.N_p), \quad (1)$$

$$N_{sp2} = N_p - N_{sp1}, \quad (2)$$

The whole population has number of solutions denoted as N_p and p denotes the ratio of N_{sp1} and N_p. In the search process of MBO, p plays significant role. Therefore, adjustment of p has its influence in two subpopulations. If p value is large, resultant solutions from N_{sp1}, otherwise from N_{sp2}. The value of p is fixed at 5/12 in the original MBO. The notion of solutions migration has its advantage as it results in novel solution. It is achieved by the application of Eq. 3 and Eq. 4.

$$x_{i,j}^{t+1} = x_{r_1,j`}^{t} \quad (3)$$

$$x_{i,j}^{t+1} = x_{r_2,j`}^{t} \quad (4)$$

The novel solutions at iteration $t+1$ are denoted as x_{r_1} and x_{r_2}. The j[th] parameter of the new solution is denoted as $x_{i,j}^{t+1}$ while $x_{r_1,j`}^{t}$ and $x_{r_2,j`}^{t}$ indicate j[th] component of the two solutions. The value of r determines whether given new solution is associated with N_{sp1} or N_{sp2} and its value is computed as in Eq. 5.

$$r = rand.peri, \quad (5)$$

The variable *peri* denotes the period of migration to control MBO functionality and *rand* indicates a number taken from a distribution. Original MBO used *peri* value as 1.2. If $r \leq p$, Eq. 3 is used to create j[th] component, otherwise Eq. 4 is used. With respect to solutions adjustment is designed for exploration and exploitation. Here search process direction relies on a pseudo-random number, denoted as *rnd*, and there are two distinguished cases. With respect to first case, if $rnd \leq p$, the current solution is adjusted to achieve a new solution as in Eq. 6.

$$x_{i,j}^{t+1} = x_{best,j`}^{t} \quad (6)$$

The j[th] parameter of the novel solution is denoted as $x_{i,j}^{t+1}$ while the current best solution's j[th] parameter associated with whole population is denoted as $x_{best,j`}^{t}$. If $rnd > p$, new solution is obtained through Eq. 7.

$$x_{i,j}^{t+1} = x_{r_3,j`}^{t} \quad (7)$$

The randomly chosen solution is denoted as r_3 which comes from N_{sp} and its j[th] parameter is denoted as $x_{r_3,j`}^{t}$. On establishing $rnd > ar$, new solutions are updated as in Eq. 8.

$$x_{i,j}^{t+1} = x_{i,j}^{t+1} + \alpha \times \left(d_{x_j} - 0.5\right) \quad (8)$$

The adjusting rate is denoted as *ar* and dx denotes a walk step, considering Lévy flights formula as expressed in Eq. 9.

$$dx = Levy(x_j^t) \quad (9)$$

In Eq. 8, α denotes a weighting factor which is computed as in Eq. 10.

$$\alpha = S_{max}/t^2, \quad (10)$$

The maximum value for walk step is denoted by S_{max} in which, a given solution moves one step at a time and the current iteration is denoted by t. The parameter α is used to exploit trade-off between exploration and exploitation. Based on its value search step becomes longer or shorter. The original MBO algorithm has significant drawbacks. First, it has insufficient power in the process of exploration as it relies on weighting factor. Second, it has limitations in striking balance between exploration and exploitation.

4. MULTI-STRATEGY BASED FUZZY ENHANCED MONARCH BUTTERFLY OPTIMIZATION ALGORITHM

This section presents the proposed solution to the problem of efficient task scheduling and load

balancing in cloud. It throws light on the proposed system model, problem context, proposed algorithm and performance evaluation metrics.

4.1 System Model

In cloud computing environment, number physical machines exist with underlying Virtual Machines (VMs). Users of the cloud across the globe send their tasks to cloud where the tasks are to be executed by allocating them to appropriate resources. The general approach in task scheduling in cloud is shown in Figure 1. The tasks submitted by ender users are kept in a queue and they are scheduled to VMs based on their availability, load and probably other factors. This is where the task scheduling algorithms play vital role.

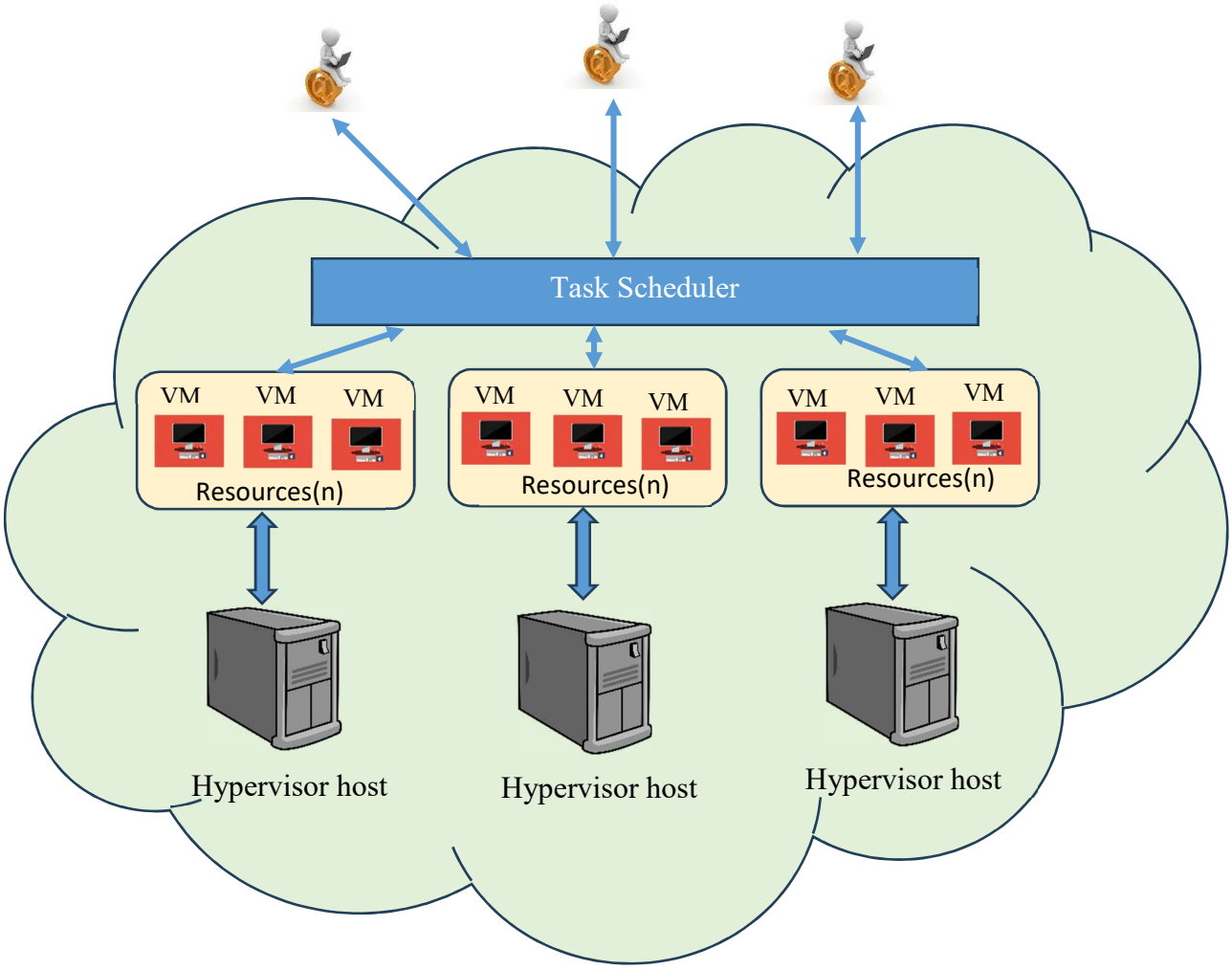

Figure 1: General approach to task scheduling in cloud

The VMs are important resources in cloud where tasks are actually executed. Hypervisor is the software in host or physical machine which enables creation of number of VMs through virtualization technology. With such technology in place, cloud resources are made affordable as it enables reduction of cost in rendering cloud computing services. When the tasks are allocated to VMs appropriately, it has many benefits such as honoring SLAs, resource optimization, load balancing and consumer satisfaction. Having understood this, Figure 2 shows our system model on top of which our algorithm is implemented using CloudSim framework

.

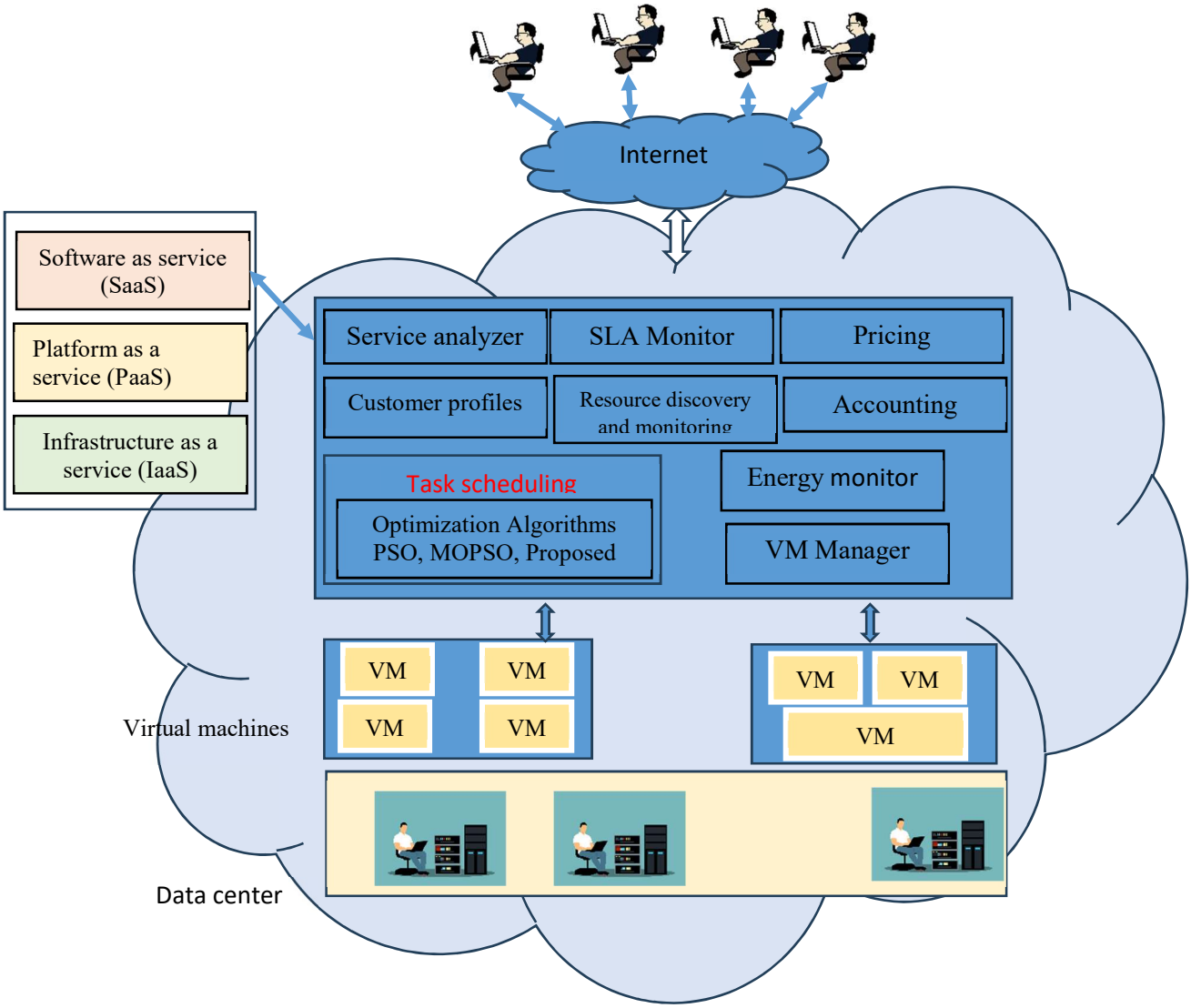

Figure 2: *The system model used for proposed task scheduling algorithm*

In cloud computing, data center plays crucial role as it provides complete infrastructure required for rendering different services to consumers. It has thousands of physical machines on top of which VMs are created to serve user requests. PaaS, SaaS and IaaS are the three service layers in cloud to provide specific services. Users of the cloud need to interact with cloud and Internet connectivity lets them to achieve this. There are many components involved in the cloud to render various services. Service analyzer takes care of service analysis. SLA monitor is meant for monitoring SLAs and their violations. Pricing and accounting components are related to pay per use model and account services respectively. Energy monitor is meant for monitoring energy consumption dynamics. Resource discovery and monitoring component is used for dealing with resources and their utilization. VM manager is meant for VM management activities while task scheduling component is used for scheduling of tasks. This is the problem context where task scheduling is the main problem considered in this paper.

4.2 Problem Definition

User tasks in cloud should be executed without violating Service Level Agreements (SLAs). SLAs are certain agreements between service provider and consumer with respect to expected services. Efficient scheduling of tasks in cloud, in presence of millions of users and diversified workloads across the globe, is NP-hard problem. Considering n number of tasks, denoted as T_i, that are independent of each other, where i= {1,2,

3…n}. There is a set of resources in cloud, that are heterogeneous in nature, denoted as R_j where j= {1, 2, 3, …, m}. The execution time for given task i on resource j is denoted as $P_{i,j}$. At any time one resource can execute only one task. Satisfying this condition, a permutation matrix, denoted as $X_{i,j}$, can be constructed where each value can be either 0 or 1 indicating a resource is free or already task allocated respectively. The permutation matrix is expressed as in Eq. 11 and Eq. 12.

$$X_{i,j} = \begin{cases} 1, & if\ task\ j\ is\ assigned\ to\ resource\ i \\ 0 & otherwise \end{cases} \quad (11)$$

$$\sum_{i=1}^{m} X_{i,j} = 1, \quad 1 \le j \le n \quad (12)$$

A resource needs certain time to executed all allocated tasks. This completion time is computed as in Eq. 13.

$$C_j = \sum_{i\in\{1\ldots n_j\}} P_{i,j} * X_{i,j} \quad (13)$$

For a given resource R_j, the number of tasks assigned is n_j and C_j indicates the completion time of tasks execution. An important objective of any task scheduling algorithm is to minimize makespan, as in Eq. 14, which indicates finish time executing all tasks.

$$Minimize\ MS = max_{1\le j\le m} C_j \quad (14)$$

While executing task assigned to a resource, there may be hardware failure which should not affect other resources. In the context, the probability with which all tasks are completed is expressed in Eq. 15.

$$p_{succ}^{i} = e^{-\lambda_i C_i} \quad (15)$$

As a failure is not dependent on other failures, for a set of resources, their ability to reliably execute tasks is expressed in Eq. 16 and at the same time failure probability needs to be minimized as expressed in Eq. 17.

$$Maximize\ R_s = e^{-\lambda_i \sum_i C_i} \quad (16)$$

$$Minimize\ F_p = \lambda_i \sum_i C_i \quad (17)$$

Here are multiple problems are such as minimizing failure probability, maximizing reliability and minimizing makespan. There is resource heterogeneity in cloud that needs to be consider in order to optimize resource utilization and balance load among resources. Therefore, load balancing is another objective function considered in the proposed system which is expressed as in Eq. 18 taken from [42].

$$\beta = \sqrt{\frac{\sum_{i=1}^{m} L_i - \bar{L}}{m}} \quad (18)$$

Where β denotes load balancing index, L denotes average load of resources and L_i denotes load of given resource. Task scheduling in cloud has its impact on resource utilization and balancing load as well.

4.3 Proposed Algorithm

We proposed an algorithm to address problems associated with original MBO by considering multiple objective functions and strategies. The design goals of the proposed algorithm are as follows. First, to enhance MBO functionality to leverage task scheduling performance. Second, to improve balance in exploration and exploitation through self-adaptive strategy. Third, application of greedy strategy towards faster convergence. Fourth, the usage of fuzzy dominance approach towards better solution selection from a set of solutions. Since our algorithm is an extension to MBO, it makes use of two swarms denoted as N_{sa} and N_{sb}. The number of individuals in populations is computed as ceil(r*N_s) and N_s-N_{sa}. Here r denotes radio of individuals in N_{sa} while N_s denotes individuals in both N_{sa} and N_{sb}. For the migration purpose variable v is computed as in Eq. 19.

$$v = rand * mp \quad (19)$$

Where *rand* indicates a random number obtained from a uniform distribution while the migration period id denoted as *mp*. We employ self-adaptive strategy explored in [43] to deal with enhancing search capability. A dynamic technique, denoted as dt, is used for self-adaptive strategy as in Eq. 20.

$$r = c + dt \quad (20)$$

Where c and d are constants generated using Eq. 21 and Eq. 22 while t indicates current iteration.

$$c = \frac{r_{min} t_m - r_{max}}{t_m - 1} \quad (21)$$

$$d = \frac{r_{max} - r_{min}}{t_m - 1} \quad (22)$$

Where maximum iteration is denoted by t_m. The minimum and maximum values for variable r lie in [0, 1] range. Update of r occurs in a linear fashion from min to max value. In the original MBO algorithm population (butterflies) are generated afresh. Due to performance issues with this approach, there is possibility of reducing convergence speed. To overcome this limitation, we used greedy strategy explored in [43] where upcoming population is chosen based on higher fitness. The greedy strategy is as expressed in Eq. 23.

$$P_{i,new}^{(t+1)} = \begin{cases} P_i^{t+1}, & f(P_i^{t+1}) < f(P_i^t) \\ P_i^t, & else \end{cases} \quad (23)$$

Here P_i^t and P_i^{t+1} are individuals while their fitness values are denoted as $f(P_i^{t+1})$ and $f(P_i^t)$ respectively. The resultant individual, denoted as $P_{i,new}^{(t+1)}$, is considered for the next iteration. We also use butterfly adjustment operator to modify location of individuals if *rand*<=*r*. The modification operation is expressed as in Eq. 24.

$$P_{m,k}^{(t+1)} = P_{best,k}^{(t)} \quad (24)$$

Where t indicates current iteration while t+1 indicates the next iteration. $P_{best,k}^{(t)}$ indicates best individual's k^{th} component in the current iteration among the two population while $P_{m,k}^{(t+1)}$ denotes position of m^{th} individual's k^{th} component. If the *rand* value is greater than *r*, then, the modification is done as in Eq. 25.

$$P_{m,k}^{(t+1)} = P_{v3,k}^{(t)} \quad (25)$$

With respect to adjustment rate, denoted as α, if rand> α then, Eq. 26 is used to modify m^{th} individual's position.

$$P_{m,k}^{(t+1)} = P_{m,k}^{(t+1)} + \beta * (dp_k - 0.5) \quad (26)$$

With the usage of waiting factor, denoted as β, in the current iteration, it is computed as in Eq. 27.

$$\beta = sw_{max}/t^2 \quad (27)$$

The highest step walk is denoted as sw_{max} used by an individual for a specific step. The step walk for m^{th} individual is computed as in Eq. 28 with the help of Levy weight.

$$dp = Levy(P_m^t) \quad (28)$$

The step size depends on β. Higher its values lead to lengthy step and lower its value leads to reduced step size. This will help in enhancing the capability of the proposed algorithm to deal with balancing exploration and exploitation. With these strategies we improved MBO and proposed an algorithm named Multi-Strategy based Fuzzy Enhanced Monarch Butterfly Optimization Algorithm (MS-FEMB0A).

Algorithm: Multi-Strategy based Fuzzy Enhanced Monarch Butterfly Optimization Algorithm

Inputs:
Number of tasks n
Number of VMs m
Iteration counter c
Maximum iteration maxIter
Output:
Efficient task scheduling and load balancing

1. Begin
2. Initialize swarm in N_{sa} and N_{sb}
3. Assess fitness function of individuals in N_{sa} and N_{sb}
4. Use fuzzy dominance to sort individuals in N_{sa} and N_{sb}
5. while c<maxIter
6. Order individuals in entire population using fitness value
7. Split them into N_{sa} and N_{sb}
8. For each individual in N_{sa}
9. Compute r using self-adaptive approach
10. Use migration operator to have new population from N_{sa}
11. Apply greedy strategy to optimize population
12. End For
13. For each individual in N_{sb}
14. Use migration operator to have new population from N_{sb}
15. End For
16. Merge individuals of N_{sa} and N_{sb}

17. Assess individuals using fitness function
18. Use fuzzy dominance for sorting
19. Keep solution in archive
20. c=c+1
21. Return solutions
22. End

Algorithm 1: Multi-Strategy based Fuzzy Enhanced Monarch Butterfly Optimization Algorithm

As presented in Algorithm 1, it takes number of tasks n, number of VMs m, iteration counter c and maximum iteration maxIter as inputs and produces solutions that can help in mapping tasks to resources for efficient scheduling and load balancing.

4.4 Simulation Parameters

The proposed system with underlying algorithm MS-FEMB0A needs number of parameters for its functionality. After experimental study, the parameter values are fixed the values that could provide best performance.

Table 1: *Shows different parameters and values used in the proposed system*

Parameter	Value
swmax	1.0
mp	1.2
v	5/12
α	5/12
maxIter	200
N_{sa}	21
N_{sb}	29

As presented in Table 1, the simulation parameters used in the proposed system are provided along with the values set in the empirical study.

5. RESULTS AND DISCUSSION

Our algorithm is implemented using Java language. It is implemented on top of CloudSim [44] for modelling and simulation of the proposed algorithm. CloudSim provides API required to model various components associated with the system model illustrated in Figure 2. In the simulation study two data centres are used with three hosts in each data centre. The hosts are configured to be heterogeneous with 2 and 4 cores. Each host is configured with 16 GB RAM and 5 VMs are associated with each host. Two datasets explored in [45] and [46] are used to evaluate the proposed algorithm. Performance of the proposed algorithm is compared with existing methods such as PSO [47] and MOPSO [48].

Table 2: *Makespan performance comparison with use of 10 resources*

Number of Tasks	Makespan		
	MS-FEMBOA	MOPSO	PSO
50	400	410	410
100	420	430	435
150	450	460	465
200	480	490	495
250	520	525	535
300	550	555	570
350	600	610	630
400	650	660	690
450	700	720	770
500	720	750	820
550	760	780	810
600	800	830	860

As presented in Table 2, performance of the proposed and existing methods is provided in terms of makespan against given number of tasks.

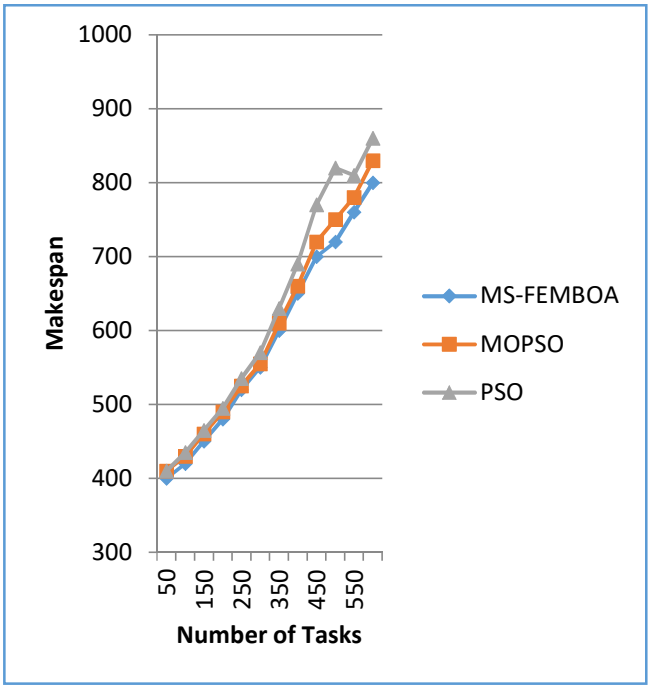

Figure 3: *Makespan performance analysis against number of tasks using 10 resources*

As presented in Figure 3, performance of different methods used for task scheduling is provided in terms of makespan. The less value for makespan measure indicates better performance. There are 12 experiments made with different number of

tasks starting with 50 and incremented by 50 each time reaching up to 600 tasks. An important observation is that number of tasks has its impact on makespan. As the number of tasks increases, the makespan also linearly increased. When number of tasks is 50, MOPSO achieved 410 as makepan, PSO 410 and proposed algorithm MS-FEMBOA achieved 400. When the number of tasks is 600, MOPSO achieved 830 as makepan, PSO 860 and proposed algorithm MS-FEMBOA achieved 800. From the results, it is observed that MS-FEMBOA outperforms the existing methods in terms of makespan.

Table 3: *Failure Probability Rate Of Scheduling Methods Against Number Of Resources*

Number of Resources	Failure Probability Rate		
	MS-FEMBOA	PSO	MOPSO
10	0.0047	0.006	0.005
20	0.004	0.0057	0.0049
30	0.0039	0.0055	0.0048
40	0.0036	0.0053	0.0044
50	0.0034	0.0054	0.0046
60	0.0031	0.005	0.0045
70	0.0033	0.0048	0.0043
80	0.0028	0.0046	0.0039
90	0.0025	0.0044	0.0037
100	0.0022	0.0042	0.0035

As presented in Table 3, failure probability rate of different scheduling methods is provided against number of resources.

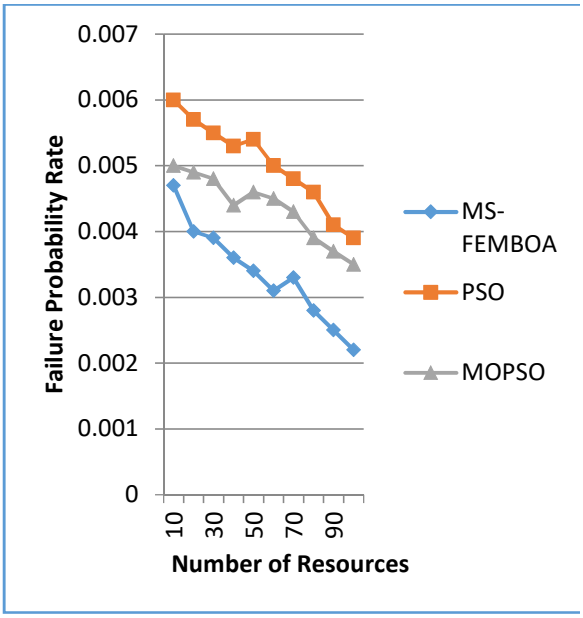

Figure 4: *Comparison Of Failure Probability Rate Of Scheduling Methods Against Number Of Resources*

As presented in Figure 4, performance of different methods used for task scheduling is provided in terms of failure probability rate. The less value for this measure indicates better performance. There are 10 experiments made with different number of resources starting with 10 and incremented by 10 each time reaching up to 100 resources. An important observation is that number of resources has its impact on failure probability. As the number of resources increases, the failure probability is linearly decreased. When number of resources is 10, PSO exhibited 0.006 as failure probability, MOPSO 0.005 and proposed algorithm MS-FEMBOA showed 0.0047 as failure probability. When the number of resources is 100, PSO exhibited 0.0042 as failure probability, MOPSO 0.0035 and proposed algorithm MS-FEMBOA achieved 0.0022. From the results, it is observed that MS-FEMBOA outperforms the existing methods in terms of failure probability rate.

Table 4: *Failure Probability Rate Of Scheduling Methods Against Number Of Tasks*

Number of Tasks	Failure Probability Rate		
	MS-FEMBOA	PSO	MOPSO
10	0.0019	0.0021	0.0025
20	0.0023	0.0028	0.0027
30	0.0026	0.0029	0.0026
40	0.0027	0.0031	0.0028
50	0.0028	0.0037	0.0036
60	0.0029	0.0038	0.0037
70	0.0032	0.0041	0.0039
80	0.0033	0.0043	0.0041
90	0.0034	0.0045	0.0043
100	0.0035	0.0048	0.0045

As presented in Table 4, failure probability rate of different scheduling methods is provided against number of tasks.

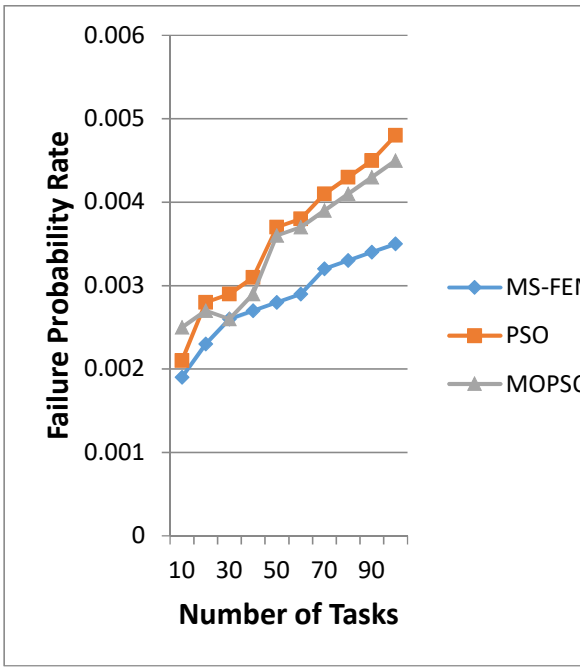

Figure 5: Comparison Of Failure Probability Rate Of Scheduling Methods Against Number Of Tasks

As presented in Figure 5, performance of different methods used for task scheduling is provided in terms of failure probability rate against number of tasks. The less value for this measure indicates better performance. There are 10 experiments made with different number of tasks starting with 10 and incremented by 10 each time reaching up to 100 tasks. An important observation is that number of tasks has its impact on failure probability. As the number of tasks increases, the failure probability is linearly increased. When number of tasks is 10, PSO exhibited 0.0021 as failure probability, MOPSO 0.0025 and proposed algorithm MS-FEMBOA showed 0.0019 as failure probability. When the number of tasks is 100, PSO exhibited 0.0048 as failure probability, MOPSO 0.0045 and proposed algorithm MS-FEMBOA achieved 0.0035. From the results, it is observed that MS-FEMBOA outperforms the existing methods in terms of failure probability rate against number of tasks.

Table 5: Convergence Of Different Methods For Makespan

Number of Iterations	Makespan (Convergence Speed Analysis)		
	MS-FEMBOA	PSO	MOPSO
0	630	795	700
200	520	635	530
400	380	635	480
600	380	600	470
800	380	600	430
1000	380	520	410
1200	380	510	410
1400	380	500	410
1600	380	500	410
1800	380	500	410
2000	380	500	410

As presented in Table 5, makespan of different scheduling methods is provided against number of iterations reflecting convergence speed.

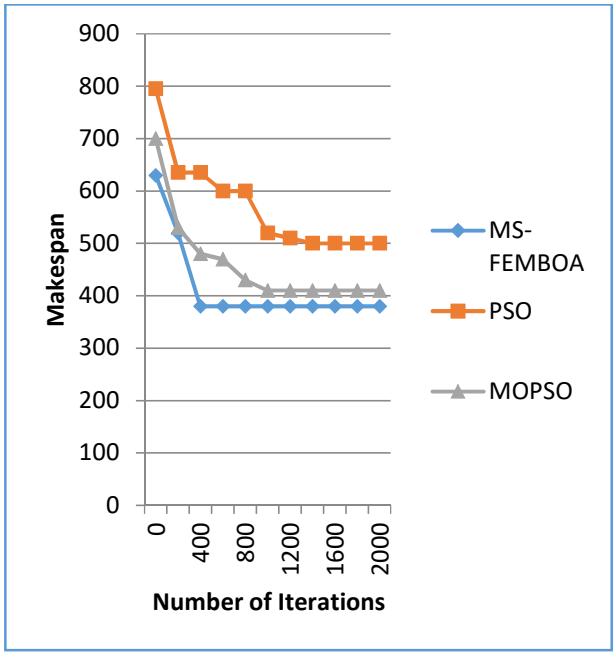

Figure 6: Comparison Of Makespan Of Scheduling Methods Against Number Of Iterations Reflecting Convergence Speed

As presented in Figure 6, performance of different methods used for task scheduling is provided in terms of makespan against number of iterations. The less value for this measure indicates better performance. There are 11 experiments made with different number of iterations starting with 0 and incremented by 200 each time reaching up to 2000 iterations. An important observation is that

number of iterations has its impact on makespan. As the number of iterations increases, the makespan is linearly decreased. When number of iterations is 0, PSO exhibited 795 as makespan, MOPSO 700 and proposed algorithm MS-FEMBOA showed 6630 makespan. When the number of tasks is 2000, PSO exhibited 500 as makespan, MOPSO 410 and proposed algorithm MS-FEMBOA achieved 380. From the results, it is observed that MS-FEMBOA outperforms the existing methods in terms of makespan against number of iterations reflecting its faster convergence.

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed an algorithm known as Multi-Strategy based Fuzzy Enhanced Monarch Butterfly Optimization Algorithm (MS-FEMB0A) for efficient task scheduling in cloud leading to load balancing. MS-FEMB0A considers multiple objectives and also multiple strategies such as fuzzy, greedy and self-adaptive towards improving search capability and convergence speed. The design goals of the proposed algorithm are as follows. First, to enhance MBO functionality to leverage task scheduling performance. Second, to improve balance in exploration and exploitation through self-adaptive strategy. Third, application of greedy strategy towards faster convergence. Fourth, the usage of fuzzy dominance approach towards better solution selection from a set of solutions. Simulation study made using CloudSim framework showed that MS-FEMB0A outperforms existing algorithms such as PSO and MOPSO. In future we intend to explore alternative scheduling approaches for task scheduling and load balancing. Towards this end, we intend to develop an evolutionary algorithm based on stochastic systems theory that exploits random variables involved in the cloud infrastructure for scheduling and load balancing.

REFERENCES

[1] Boutaba, Raouf; Salahuddin, Mohammad A.; Limam, Noura; Ayoubi, Sara; Shahriar, Nashid; Estrada-Solano, Felipe and Caicedo, Oscar M. (2018). A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. Journal of Internet Services and Applications, 9(1), 16–. http://doi:10.1186/s13174-018-0087-2

[2] Bali, Kavitesh Kumar; Ong, Yew-Soon; Gupta, Abhishek and Tan, Puay Siew (2019). Multifactorial Evolutionary Algorithm with Online Transfer Parameter Estimation: MFEA-II. IEEE Transactions on Evolutionary Computation, 1–1. http://doi:10.1109/TEVC.2019.2906927

[3] Darwish, Ashraf; Hassanien, Aboul Ella and Das, Swagatam (2019). A survey of swarm and evolutionary computing approaches for deep learning. Artificial Intelligence Review. http://doi:10.1007/s10462-019-09719-2

[4] Drugan, Mădălina M. (2018). Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. Swarm and Evolutionary Computation, S2210650217302766–. http://doi:10.1016/j.swevo.2018.03.011

[5] Zhao, Haitong and Zhang, Changsheng (2020). An online-learning-based evolutionary many-objective algorithm. Information Sciences, 509, 1–21. http://doi:10.1016/j.ins.2019.08.069

[6] Parisi, Luca and RaviChandran, Narrendar (2020). Evolutionary Denoising-Based Machine Learning for Detecting Knee Disorders. Neural Processing Letters. http://doi:10.1007/s11063-020-10361-1

[7] Wei, Leyi; Tang, Jijun and Zou, Quan (2016). Local-DPP: An Improved DNA-binding Protein Prediction Method by Exploring Local Evolutionary Information. Information Sciences, S0020025516304509–. http://doi:10.1016/j.ins.2016.06.026

[8] Assunção, Filipe; Lourenço, Nuno; Machado, Penousal and Ribeiro, Bernardete (2018). DENSER: deep evolutionary network structured representation. Genetic Programming and Evolvable Machines. http://doi:10.1007/s10710-018-9339-y

[9] Zhang, Lu; Tan, Jianjun; Han, Dan and Zhu, Hao (2017). From machine learning to deep learning: progress in machine intelligence for rational drug discovery. Drug Discovery Today, 22(11), 1680–1685. http://doi:10.1016/j.drudis.2017.08.010

[10] Kabra, Ritika and Singh, Shailza (2021). Evolutionary artificial intelligence based peptide discoveries for effective Covid-19 therapeutics. Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease, 1867(1), 165978–. http://doi:10.1016/j.bbadis.2020.165978

[11] Sahoo, Kshira Sagar; Tripathy, Bata Krishna; Naik, Kshirasagar; Ramasubbareddy,

Somula; Balusamy, Balamurugan; Khari, Manju and Burgos, Daniel (2020). An Evolutionary SVM Model for DDOS Attack Detection in Software Defined Networks. IEEE Access, 8, 132502–132513. http://doi:10.1109/ACCESS.2020.3009733

[12] Aziza, Hatem and Krichen, Saoussen (2020). A hybrid genetic algorithm for scientific workflow scheduling in cloud environment. Neural Computing and Applications. http://doi:10.1007/s00521-020-04878-8

[13] Ismayilov, Goshgar and Topcuoglu, Haluk Rahmi (2020). Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. Future Generation Computer Systems, 102, 307–322. http://doi:10.1016/j.future.2019.08.012

[14] Pławiak, Paweł (2017). Novel Methodology of Cardiac Health Recognition Based on ECG Signals and Evolutionary-Neural System. Expert Systems with Applications, S0957417417306292–. http://doi:10.1016/j.eswa.2017.09.022

[15] Onan, Aytuğ; Korukoğlu, Serdar and Bulut, Hasan (2017). A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification. Information Processing & Management, 53(4), 814–833. http://doi:10.1016/j.ipm.2017.02.008

[16] Jin, Yaochu; Wang, Handing; Chugh, Tinkle; Guo, Dan and Miettinen, Kaisa (2018). Data-Driven Evolutionary Optimization: An Overview and Case Studies. IEEE Transactions on Evolutionary Computation, 1–1. http://doi:10.1109/TEVC.2018.2869001

[17] Hu, Ya-Lan and Chen, Liang (2018). A nonlinear hybrid wind speed forecasting model using LSTM network, hysteretic ELM and Differential Evolution algorithm. Energy Conversion and Management, 173, 123–142. http://doi:10.1016/j.enconman.2018.07.070

[18] Peng, Lu; Liu, Shan; Liu, Rui and Wang, Lin (2018). Effective Long short-term Memory with Differential Evolution Algorithm for Electricity Price Prediction. Energy, S0360544218308727–. http://doi:10.1016/j.energy.2018.05.052

[19] Wang, Handing; Jin, Yaochu; Sun, Chaoli and Doherty, John (2018). Offline Data-Driven Evolutionary Optimization Using Selective Surrogate Ensembles. IEEE Transactions on Evolutionary Computation, 1–1. http://doi:10.1109/TEVC.2018.2834881

[20] Tuan, Tong Anh; Long, Hoang Viet; Son, Le Hoang; Kumar, Raghvendra; Priyadarshini, Ishaani and Son, Nguyen Thi Kim (2019). Performance evaluation of Botnet DDoS attack detection using machine learning. Evolutionary Intelligence. http://doi:10.1007/s12065-019-00310-w

[21] Faris, Hossam; Al-Zoubi, Ala' M.; Heidari, Ali Asghar; Aljarah, Ibrahim; Mafarja, Majdi; Hassonah, Mohammad A. and Fujita, Hamido (2018). An Intelligent System for Spam Detection and Identification of the most Relevant Features based on Evolutionary Random Weight Networks. Information Fusion, S1566253518303968–. http://doi:10.1016/j.inffus.2018.08.002

[22] J. Carrasco, S. García, M.M. Rueda, S. Das and F. Herrera. (2020). Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guide. *Elsevier*. 54, pp.1-52. https://doi.org/10.1016/j.swevo.2020.100665

[23] Praveen Kumar, D.; Amgoth, Tarachand and Annavarapu, Chandra Sekhara Rao (2019). Machine learning algorithms for wireless sensor networks: A survey. Information Fusion, 49, 1–25. http://doi:10.1016/j.inffus.2018.09.013

[24] Abba, S.I.; Hadi, Sinan Jasim; Sammen, Saad Sh.; Salih, Sinan Q.; Abdulkadir, R.A.; Pham, Quoc Bao; Yaseen, Zaher Mundher (2020). Evolutionary computational intelligence algorithm coupled with self-tuning predictive model for water quality index determination. Journal of Hydrology, 587(), 124974–. doi:10.1016/j.jhydrol.2020.124974

[25] Ben Chaabene, Wassim; Flah, Majdi and Nehdi, Moncef L. (2020). Machine learning prediction of mechanical properties of concrete: Critical review. Construction and Building Materials, 260, 119889–. http://doi:10.1016/j.conbuildmat.2020.119889

[26] Heaton, Jeff (2017). Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. Genetic Programming and Evolvable Machines, s10710-017-9314-z–. http://doi:10.1007/s10710-017-9314-z

[27] Feng, Liang; Zhou, Lei; Zhong, Jinghui; Gupta, Abhishek; Ong, Yew-Soon; Tan, Kay-Chen and Qin, A. K. (2018). Evolutionary Multitasking via Explicit Autoencoding. IEEE Transactions on Cybernetics, 1–14. http://doi:10.1109/TCYB.2018.2845361

[28] Wu, Xueqi and Che, Ada (2018). A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. Omega, S0305048317307922–. http://doi:10.1016/j.omega.2018.01.001

[29] Preksha Pareek and Ankit Thakkar; (2021). RGB-D based human action recognition using evolutionary self-adaptive extreme learning machine with knowledge-based control parameters . Journal of Ambient Intelligence and Humanized Computing. http://doi:10.1007/s12652-021-03348-w

[30] Su, Jiawei; Vargas, Danilo Vasconcellos and Sakurai, Kouichi (2019). One Pixel Attack for Fooling Deep Neural Networks. IEEE Transactions on Evolutionary Computation, 1–1. http://doi:10.1109/TEVC.2019.2890858

[31] Shi, Han; Liu, Simin; Chen, Junqi; Li, Xuan; Ma, Qin and Yu, Bin (2018). Predicting drug-target interactions using Lasso with random forest based on evolutionary information and chemical structure. Genomics, S088875431830466X–. http://doi:10.1016/j.ygeno.2018.12.007

[32Fernandez, Alberto; Herrera, Francisco; Cordon, Oscar; Jose del Jesus, Maria and Marcelloni, Francesco (2019). Evolutionary Fuzzy Systems for Explainable Artificial Intelligence: Why, When, What for, and Where to?. IEEE Computational Intelligence Magazine, 14(1), 69–81. http://doi:10.1109/MCI.2018.2881645

[33] Chen, Wei; Panahi, Mahdi; Tsangaratos, Paraskevas; Shahabi, Himan; Ilia, Ioanna; Panahi, Somayeh; Li, Shaojun; Jaafari, Abolfazl and Ahmad, Baharin Bin (2019). Applying population-based evolutionary algorithms and a neuro-fuzzy system for modeling landslide susceptibility. CATENA, 172, 212–231. http://doi:10.1016/j.catena.2018.08.025

[34] Vivekanandan, T. and Sriman Narayana Iyengar, N. Ch (2017). Optimal feature selection using a modified differential evolution algorithm and its effectiveness for prediction of heart disease. Computers in Biology and Medicine, 90, 125–136. http://doi:10.1016/j.compbiomed.2017.09.011

[35] Jiménez, F.; Sánchez, G.; García, J.M.; Sciavicco, G. and Miralles, L. (2017). Multi-objective evolutionary feature selection for online sales forecasting. Neurocomputing, 234, 75–92. http://doi:10.1016/j.neucom.2016.12.045

[36] Taradeh, Mohammad; Mafarja, Majdi; Heidari, Ali Asghar; Faris, Hossam; Aljarah, Ibrahim; Mirjalili, Seyedali and Fujita, Hamido (2019). An Evolutionary Gravitational Search-based Feature Selection. Information Sciences, S0020025519304414–. http://doi:10.1016/j.ins.2019.05.038

[37] Pan, Linqiang; He, Cheng; He, Cheng; Tian, Ye; Wang, Handing; Zhang, Xingyi and Jin, Yaochu (2018). A Classification Based Surrogate-Assisted Evolutionary Algorithm for Expensive Many-Objective Optimization. IEEE Transactions on Evolutionary Computation, 1–1. http://doi:10.1109/TEVC.2018.2802784

[38] Kumar, Jitendra and Singh, Ashutosh Kumar (2018). Workload prediction in cloud using artificial neural network and adaptive differential evolution. Future Generation Computer Systems, 81, 41–52. http://doi:10.1016/j.future.2017.10.047

[39] Yang, Li and Shami, Abdallah (2020). On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice. Neurocomputing, S0925231220311693–. http://doi:10.1016/j.neucom.2020.07.061

[40] Yunlang Xu; Xiaofeng Yang; Zhile Yang; Xiaoping Li; Pang Wang; Runze Ding and Weike Liu; (2021). An enhanced differential evolution algorithm with a new oppositional-mutual learning strategy . Neurocomputing. http://doi:10.1016/j.neucom.2021.01.003

[41] Wang, G.G.; Deb, S.; Cui, Z. Monarch Butterfly Optimization. Neural Comput. Appl. 2015.

[42] Palos-Sanchez, P.R.; Arenas-Marquez, F.J.; Aguayo-Camacho, M. Cloud Computing (SaaS) Adoption as a Strategic Technology: Results of an Empirical Study. Mob. Inf. Syst. 2017, 2017.

[43] Strumberger, I.; Tuba, E.; Bacanin, N.; Beko, M.; Tuba, M. Monarch butterfly optimization algorithm for localization in wireless sensor networks. In Proceedings of the 2018 28th International Conference

Radioelektronika (RADIOELEKTRONIKA), Prague, Czech Republic, 19–20 April 2018; pp. 1–6.

[44] Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.F.; Buyya, R. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Softw. Pract. Exp. 2011, 41, 23–50.

[45] M. Sivaram, D. Yuvaraj, A. S. Mohammed, V. Manikandan, V. Porkodi et al., "Improved enhanced DBTMA with contention-aware admission control to improve the network performance in MANET S," computers, Materials & Continua, vol. 60, no. 2, pp. 435W54, 2019.

[46] V. Manikandan, M. Sivaram, A. S. Mohammed and V. Porkodi, "Nature inspired improved firefly a lgorithm for node clustering in WSNs," computers, Materials & Continua, vol. 64, no. 2, pp. 753-7 76, 2020.

[47] Huang, X., Li, C., Chen, H. et al. Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies. Cluster Comput 23, 1137–1147 (2020). https://doi.org/10.1007/s10586-019-02983-5

[48] Alkayal, Entisar S.; Jennings, Nicholas R.; Abulkhair, Maysoon F. (2016). 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops) - Efficient Task Scheduling Multi-Objective Particle Swarm Optimization in Cloud Computing. , (), 17–24. doi:10.1109/LCN.201 6.024