Received: April 11, 2019

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Resource Allocation Using Democratic Grey Wolf Optimization in Cloud Computing Environment

Alok Kumar Pani[1]* **Bhawna Dixit[1]** **Kailash Patidar[1]**

[1]*Department of Computer Science and Engineering,*
Sri Satya Sai University of Technology and Medical Sciences, India
* Corresponding author's Email: alok.kumar.pani@gmail.com

Abstract: Cloud computing is one of the growing environment that outlines an internet-based computing system consisting of a large number of computers and other devices, where a number of things, such as computer infrastructure, access to applications, software, processing power, etc. are shared over the Internet. The allocation of the cloud resources to the user based on their request is a Non-deterministic Polynomial-time (NP) issue, which consumes more time. Therefore, heuristic methods utilize for optimizing resource allocation. In this research work, the Democratic Grey Wolf Optimization (DGWO) is proposed to overcome the limitations of the resource allocation in efficient way. In this method, DGWO performs the following steps; initializing the request size, generating requests, and estimate fitness value of DGWO, sorting, dividing and evaluating the requests of the user. The advantages of DGWO are higher speed convergence, easier implementation, global optimization capacity. The DGWO has high performance in unknown, challenging search spaces which has high local optima avoidance. The efficiency of DGWO is improved in terms of searching optimum resource time where the searching behavior is obtained from GWO. The DGWO method achieved 75% throughput, 96% allocation of resources compared to other meta-heuristic existing techniques.

Keywords: Cloud computing, Democratic grey wolf optimization, Hardware resources, Throughput, Virtual machines.

1. Introduction

According to virtualization of resources, cloud computing becomes a trend methodology for computing [1], which provides main services such as Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) [2, 3]. Resource management has a significant effect on Resource Allocation (RA) that is the Cloud datacentres are usually deployed as private, public and or hybrid. Cloud service provider tools are used for service deployment and this can be achieved, when appropriate datacentre resources are selected to run for actual application requirement in IaaS [4]. In a cloud environment the physical machines run multiple Virtual Machines (VM) which are presented to the clients as the computing resources. In cloud environment the physical machines run multiple virtual machines (VM) which are delivered to the clients as the computing resources [5]. The optimal number of servers obtain by resource allocation techniques which is used to allocate resources based on the user's requirements. The cloud users can activate and change the resource loads with applications in minimum expenditure and rent the resources from providers [6, 7].

Every user desires multiple resources intended for a defined task that improves the performance and finished on time [8]. Cloud computing is successfully utilized by organizations as it offers extensive solutions along with increasing flexibility, scalability, agility, reduces costs and higher efficiencies [9]. In the Cloud, Resource management consists of two stages namely resource scheduling and resource provisioning [10]. The cloud's clients are interested to complete their works in the shortest possible time and at minimum cost, in addition to this, providers are interested to increase the profit by maximizing the use

of their resources with a lower overall cost [11]. These two objectives are in conflict and often they are not satisfied by using traditional resource allocation and load balancing techniques [12]. Traditional methods such as FIFO and Round-Robin is often trapped in local optimum, inaccurate, inconclusive, time consuming and also not effective for allocating the resources in multi-objective scheduling [13-15]. The meta-heuristic algorithms have a critical place in the field of research in the last few years and various meta-heuristics algorithms have some novel variations and are suggested for the allocation of resources in many fields [16].

There are many prominent meta-heuristic algorithms which are prominent as well as remarkable in cloud computing such as Ant Colony Optimization (ACO), the Cuckoo Search (CS), the Firefly Algorithm (FA), the Memetic Algorithm (MA) and so on. The objective of the paper is to implement the DGWO algorithm for resource allocation. The request size are initialized, fitness value of DGWO are calculated after the generation of requests. The DGWO achieved global optimization capacity, higher convergence speed and high performance in challenging search spaces. The implementation of DGWO is easier and have high local optima avoidance. The performance of the proposed method is compared with the existing methods which are explained in the following sections.

The draft of the paper follows: Section 2 presented a brief survey on the recent techniques of the resource allocation, Section 3 presents the problem statement of the research work, the proposed methodology is presented in Section 4. The experimental results are given in Section 5. The conclusion of the work done with future work is given in Section 6.

2. Literature review

T. Jena, and J.R. Mohanty, [17] proposed Genetic Algorithm (GA) based Customer-Conscious RA and task scheduling (GACCRATS) to fill the gap between frequently changing customer requirement and available infrastructure for services. In order to attain minimum makespan time and maximum customer satisfaction, the GA method mapped the tasks to the VM of a multi-cloud federation. The experimental analysis stated that the GA-based RA provided better performance compared to other existing heuristic algorithms. The scalability of the simulated multi-cloud environment was considerably high, so the method was unable to consider the data locality cost, energy consumption and running cost.

X. Liu, X. Zhang, W. Li, and X. Zhang [18] implemented the optimization algorithms Discrete Artificial Bee Colony (DABC), Discrete Artificial Fish Swarm (DAFS), and Discrete Shuffled Frog Leaping (DSFL) to solve the NP-hard problems. They designed the self-adaptive parameter settings to balance between the exploitation and exploration of the algorithm. The experimental analysis showed that these three optimization algorithms increased the resource utilization and maximized the global dominant share, which was highly adaptable to several situations. With the grouping strategy, the results proved that the DSFL effectively solved the local optimal problem and found better solutions, but it took more computation time.

H. Zheng, Y. Feng, and J. Tan [19] implemented the hybrid energy-aware RA approach for helping requestors to acquire energy efficient and satisfied manufacturing services. To conduct the combinatorial optimization process, Non-dominated Sorting Genetic Algorithm (NSGA-II) and multi-objective mathematical model was adopted. The method needs to study fuzzy information processing and intelligent algorithm for better performance of the NSGA-II approach.

M.H. Malekloo, N. Kara, and M. El Barachi [20] executed an energy-aware and QoS-aware Multi-Objective Ant Colony Optimization (MACO) approach for VM position and solidification. This approach aimed to obtain a tradeoff among energy proficiency, framework execution, and SLA-consistence. The experimental results stated that the MACO approach saved energy and reduced the quality of VM relocations and SLA violations. The bottleneck of this method was unpredictability of cost, and a lapse in execution because of difficulties in finding necessities.

F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, and H. Tenhunen [21] implemented a dispersed framework design to perform dynamic VM consolidation to diminish energy utilization of cloud server centers while maintaining the desired QoS. The proposed ACO-based VM Consolidation (ACOVMC) approach found a close optimal arrangement based on a predetermined target work. The method outperformed existing VM consolidation methodologies in terms of energy utilization, a number of VM movements, and QoS necessities concerning execution. The method decreased the energy utilization of data centres by consolidating VMs into the number of active Physical Machines (PM) while preserving the quality of service requirements.

P. Durgadevi, and S. Srinivasan, [22] proposed the hybridized optimization algorithm which combined Shuffled Frog Leaping Algorithm (SFLA) with CS for allocating the resource effectively. The optimization

issues were solved by using SFLA-CS with the help of preceding steps that included initialization and generation of requests. The SFLA-CS method achieved higher convergence speed with the capacity of having global optimization. The experiments were conducted to evaluate the performance of SFLA-CS with existing techniques such as Krill herd, CS. The execution time was high in SFLA-CS method when compared with other existing techniques such as ABC and CS.

3. Allocation of resource

Resource allocation techniques help to attain an optimal number of servers by allocating resources to users based on their application demands. In cloud computing, there are two major limitations for allocating the resources. First, the capacity of the machines is physically limited; second, priorities for the implementation of the tasks should be in direction with the efficiency of resources. Ultimately, the waiting time and the completion time need to be reduced, in order to decrease the cost of system implementation. Traditional approaches solve the optimization problem, but these approaches are often trapped in a local optimum, time-consuming and inaccurate.

4. Proposed methodology

The resources are allocated to users by using DGWO method, which performs several operations such as initialize the request size, generate the requests and calculate the fitness values of DGWO. Then the request of users is sorted, divided, modified the position of a request and finally evaluate the new solution for users. Then the resources are finally assigned to the cloud computing server grounded on the user requests. Finally, the DGWO algorithm diminishes the waiting time of the user request. The requests are then perfectly allocated on the cloud server. The block diagram for the technique is presented in Fig. 1.

4.1 User request

In the primary phase, the user requests are transmitted to the cloud environment. Then the users' requests are assigned using following Eq. (1),

$$UR_q(x) = \{UR_{q1}, UR_{q2}, UR_{q3}, \ldots., UR_{qn}\} \quad (1)$$

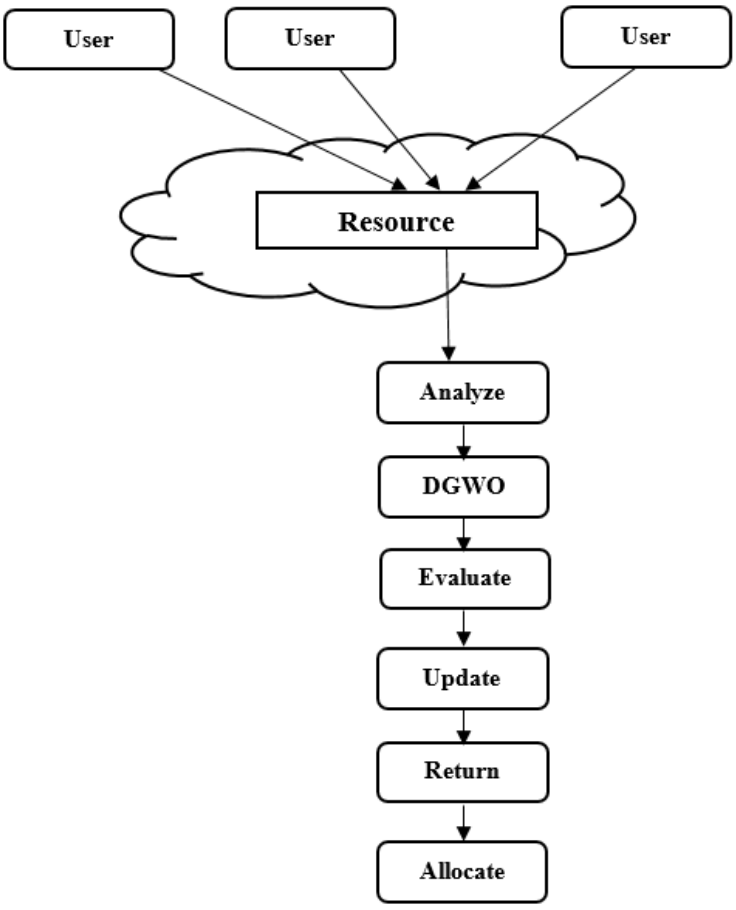

Figure.1 Block Diagram of the proposed method

Where, $U\,R_q(x)$ is Number of requests, $x = \{1, 2, 3, \ldots, n\}$. The requests are gathered from the users for every particular time interval by the real-time request queue.

4.2 Analyze request

The requests of user are analysed one by one according to the request speed, weight and energy in the secondary phase. The optimization process starts after analysing the user request, which are described in following section.

4.3 Democratic grey wolf optimization

The natural inspiration of GWO technique is the strategy of grey wolves which are belongs to family Canidae. Grey wolfs have the nature of group living and each group consists of 5-12 members, which strictly organize to a social dominant hierarchal system. Each member of the group plays a part in the hunting process. The wolfs in the groups ranked as alpha, beta, omega and remaining subordinate wolfs classified as delta. The alpha wolfs are the leader of the group and it is in top of the grey wolf's hierarchy, which takes the decision about the hunting and food.

Alpha wolves (i.e. Female Wolf) are the decision makers of the group and controlling all the living activities of the group including the hunting. Beta wolves are the subordinate to the alpha wolves, they support the decision made by the alpha wolves. The Beta wolves are the potential candidate for supporting the cause of the alpha. Omega wolves present in the next rank in the group and they maintain the group dominance hierarchal structure. The rest of the member of the group are classified as delta and they are sub ordinance to the Omega. GWO works on the basis of the hunting process of the grey wolves. The hunting process of wolf's is mathematically represented by the following phases:

1. Track and chase the prey
2. Purse, encircle and annoy the prey
3. Attach the prey.

GWO solution is divided into three-level based on the fitness and optimality of the solution. Probably the alpha's decision is the fittest solution for the optimizing problem. The beta and delta decision are ranked as the next best two solutions and the remaining solutions are by omega. The process of encircling is given in Eq. (2).

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) + \vec{A}.\vec{D} \tag{2}$$

The $\vec{X}$ in the Eq. (2) gives the location of the prey and t represents the iteration number, $\vec{A}$ and $\vec{C}$ are the coefficient vector. The value of $\vec{D}$ which is a behaviour of encircling prey is given in Eq. (3).

$$\vec{D} = |\vec{C}.\overrightarrow{X_p}(t) - \vec{X}(t)| \tag{3}$$

The $\vec{A}$ and $\vec{C}$ are manipulated in the Eq. (4) and (5).

$$\vec{A} = 2\vec{a}.\overrightarrow{r_1} - \vec{a} \tag{4}$$

$$\vec{C} = \overrightarrow{2r_2} \tag{5}$$

The components of $\vec{a}$ are decreased from 2 to 0 over the number of iteration and $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are the random vector in [0, 1].

Alpha always takes the lead for the hunting, beta and delta knows the location of the prey, this gives the best three solutions to update the location followed by other agents based on the best search agent. The Eq. for the position updating is given in the Eq. (6) to (8).

$$\overrightarrow{D_a} = \left|\overrightarrow{C_1}.\overrightarrow{X_a} - \vec{X}\right|, \overrightarrow{D_\beta} = \left|\overrightarrow{C_2}.\overrightarrow{X_\beta} - \vec{X}\right|, \overrightarrow{D_\delta} = |\overrightarrow{C_3}.\overrightarrow{X_\delta} - \vec{X}| \tag{6}$$

$$\overrightarrow{X_1} = \left|\overrightarrow{X_a} - \overrightarrow{A_a D_\alpha}\right|, \overrightarrow{X_2} = \left|\overrightarrow{X_\beta} - \overrightarrow{A_2 D_\beta}\right|, \overrightarrow{X_3} = |\overrightarrow{X_\delta} - \overrightarrow{A_3 D_\delta}| \tag{7}$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{8}$$

In GWO, the updation of $\vec{a}$ in every round [2, 0] is used to control the tradeoff between the exploration and exploitation which is given in Eq. (9).

$$\vec{a} = 2 - t.\frac{2}{Max_i ter} \tag{9}$$

Where t is the number of the iteration and $Max_i ter$ is the maximum number of iteration for optimization. By using Eq. (9), the optimal values are calculated with the help of request, and for every request, the optimal value is calculated individually and the whole values are summed to get the better solution. Finally, DGWO method allocated the resources by using entire user requests.

5. Experimental result

The time for allocation of resources, processing elements capacity and the knapsack issues in RA are reduced by using proposed work. The proposed DGWO method executed in JAVA with CloudSim tool and the efficiency of DGWO method validated in terms of execution time, allocation mechanism, throughput and so on. CloudSim is a software tool used for simulations in cloud environment based experiments. The parameters which are used in proposed DGWO is described in below Table 1.

5.1 Evaluation of parameter metrics

In this section, the performance of the proposed method is calculated by evaluating the following parameters such as execution time, throughput, time based task execution, waiting time, energy consumption, makespan and allocation mechanism. The parameters of the proposed method compared with existing methodologies such as Hybrid ABC-CS algorithm (HABBCS), krill herd and Shuffled Frog Leaping Algorithm (SFLA)- CS Algorithm. In complex situations, these existing techniques possess the request speed and sizes are increased. Those evaluations are carried out in server-side only when allocating the resources. While considering the pre-specifie tolerance, the fitness function of SFLA-CS is very less within the successive iterations. To overcome this problem, DGWO is proposed with better fitness function for allocating the resources to the users.

5.1.1. Execution time

The execution time of DGWO is very less when compared with other existing methods, and those values are tabulated in Table 1. DGWO required 3800 ms for executing the task, but Krill herd and SFLA-CS required $8500\,ms, 4500\,ms$ respectively. The execution time is computed in Eq. (10),

$$E_T = E(t) - F(t) \tag{10}$$

Where E_T is the computational time, $E(t)$ is the ending time of the process, $F(t)$ is the beginning time of the process. Table 2 and Fig. 2 represents the performance parameters such as execution time of the proposed method with existing methodologies.

5.1.2. Throughput

Throughput refers to the quantity of information transported as of one site to a dissimilar one in a specified quantity of time. It is utilized to measure the performances of hard drives, RAM, Internet and the network connections. The throughput of the proposed system should be higher than the existing system throughput. The throughput is estimated in Eq. (11).

$$T_t = \frac{I_t}{t} \tag{11}$$

Where, I_t is the transported information and t is the quantity of time. The throughput of different algorithms is compared and exhibited in Table 3 and Fig. 3.

Table 1. Parameter settings

Parameters	Value of parameters
Number of Clouds	5, 10, 15, 30, 50
Number of tasks	15,50,60,100,150,200,250,300
Structure of Datasets	(Number of tasks) * (number of virtual machines)
Reserved instances	i1, i2, i3, i4

Table 2. Performance of execution time

Methods	Execution time (ms)
HABCCS [22]	15000
Krill herd [22]	8500
SFLA-CS [22]	4500
DGWO	3800

Table 3. Performance of Throughput

Methods	Throughput (in sec)
HABCCS [22]	15
Krill herd [22]	45
SFLA-CS [22]	60
DGWO	75

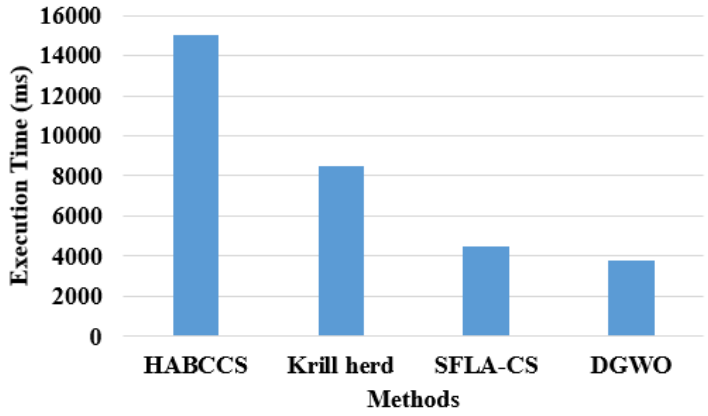

Figure.2 Performance of execution time

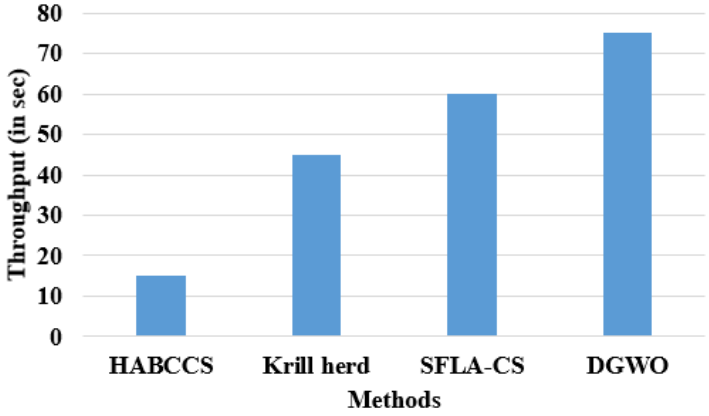

Figure.3 Performance of throughput

5.1.3. Time based task execution

The task is scheduled by time-based task execution or else event-cantered triggers are set on a task which starts its execution. The time-based task execution of DGWO are compared with HABCCS, SLFA-CS and Krill herd, and their results are tabulated in Table 3. The performance of DGWO in task-based execution is shown in Table 4.

5.1.4. Waiting time

Waiting time (Turnaround time) is defined as the time taken for processing the request, and consummation of the request which is computed using Eq. (12).

$$W(t)_{Avg} = T(t)_{Avg} - B(t) \quad (12)$$

where $W(t)_{Avg}$ is 'average waiting time', $T(t)_{Avg}$ is considered as 'average turnaround time', then $B(t)$ is 'burst time'. The comparison on the waiting time is given in Table 5.

5.1.5. Allocation mechanism

In this section, the allocation mechanism of DGWO are compared with other existing methods, which is shown in Table 6. Overall, the DGWO optimization algorithm is better in allocating resources to users in the cloud environment. Fig. 4 presents the graphical representation of the proposed method in terms of allocation mechanism.

The results show that the proposed DGWO approach performs well by achieving 75% throughput by the execution time of 4000 ms.

5.1.6. Makespan

The total execution of all task is defined as Makespan, which is explained in Eq. (13)

$$Makespan = \sum_{i=0}^{n} Executiontime(Task_i) \quad (13)$$

In this section, makespan of proposed DGWO is compared with existing method GACCRATS and Teacher Learning Based Optimization (TLBO) [17] for validating the performance of DGWO. Table 7 describes the makespan time of both existing methods and DGWO. Fig. 5 describes the graphical

representation of makespan of DGWO with existing techniques.

Table 4. Performance of Time based Evaluations

Number of Tasks	Time based Evaluations			
	HABCCS [22]	Krill herd [22]	SLFA-CS [22]	DGWO
50	60	30	18	10
200	80	47	32	20
250	75	65	50	35
300	83	94	25	15

Table 5. Performance of Waiting Time

	Waiting Time			
Number of Tasks	HABCCS [22]	Krill herd [22]	SLFA-CS [22]	DGWO
15	40	210	260	300
60	30	170	220	260
100	35	150	170	200
150	25	130	150	195

Table 6. Performance of Allocation Mechanism

Methods	Allocation (%)
HABCCS [22]	54
Krill herd [22]	75
SFLA-CS [22]	93
DGWO	96

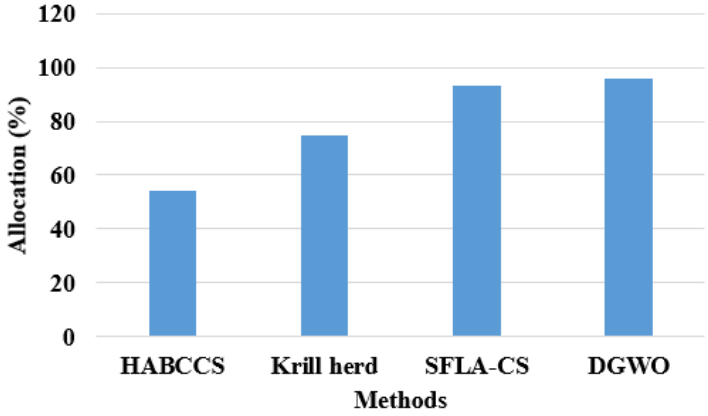

Figure.4 Performance of allocation mechanism

Table 7. Comparison of makespan of DGWO

Methods	Makespan Time
TLBO	5.004
GACCRATS	3.378
DGWO	2.781

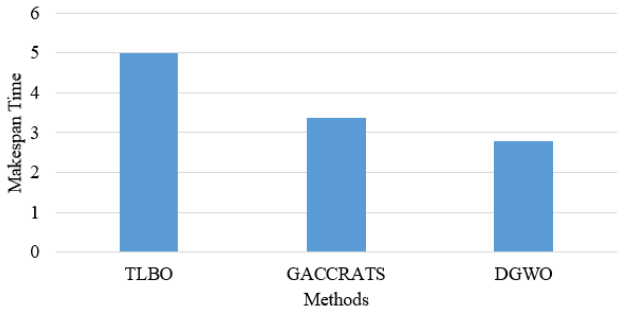

Figure.5 Makespan comparison of DGWO

From the above Table 6 and Fig. 5, the proposed DGWO method consume less makespan than all the existing methods such as TLBO and GACCRATS. Makespan is required to have a lower value. TLBO algorithm fails to achieve this goal as it takes longer time to execute, when compared with GACCRATS.

5.1.7. Energy consumption

The application workloads are handled by physical resources in a data center which consumes energy a lot. According to memory, disk, network card and utilization of CPU, the energy is consumed. Table 8 describes the consumption of energy of proposed DGWO for different tasks with existing method ACOVMC approach [21]. Fig. 6 describes the energy consumption of DGWO with ACOVMC.

From the above Table 8, it clearly shows that the proposed DGWO consumed less energy when compared with ACOVMC. The DGWO achieved 89.4% energy whereas ACOVMC achieved 110% energy for the 100th task. When the number of task increases, the consumption of energy for both methods increases. The ACOVMC achieved 129% energy for 250th task, whereas the proposed DGWO consumed nearly 20% less energy.

5.1.8. Overall performance of DGWO

In this section, the overall performance of DGWO is evaluated by using parametric such as execution time, throughput and allocation time, Time based evaluation and waiting time. Table 9 describes the

performance values of DGWO in terms of allocation

Table 8. Energy Consumption

Number of task	ACOVMC	Proposed DGWO
50	96	86
100	110	89.4
150	121	92.6
200	125	96
250	129	99.5

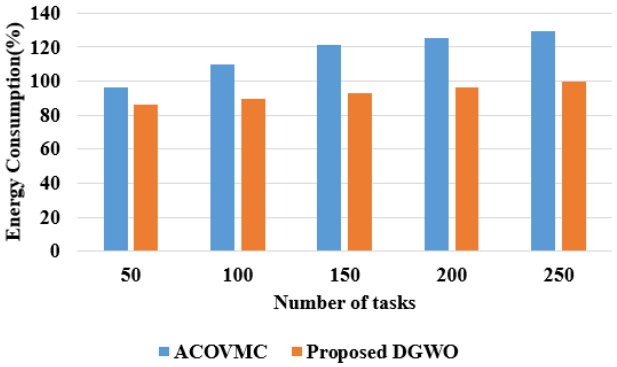

Figure.6 Energy consumption of DGWO

Table 9. Overall performance of DGWO

Methodology	Allocation Mechanism	Through put	Execution Time
HABCCS [22]	54	15	15000
SFLA-CS [22]	93	60	5000
DGWO	96	75	3800

Table 10. Performance of DGWO

Methodology	Time based Evaluation				Turnaround time			
	Number of Tasks				Number of Tasks			
	50	200	250	300	15	60	100	150
Krill herd [22]	30	47	65	94	210	170	150	130
SFLA-CS [22]	18	32	50	25	260	220	170	150
DGWO	10	20	35	15	300	260	200	195

mechanism, throughput and execution time. The values are compared with only two optimization techniques such as SFLA-CS and HABCCS.

When compared with HABCCS and SFLA-CS, the DGWO achieved higher allocation mechanism, less execution time and throughput. The DGWO achieved 96% allocation of resources whereas the HABCSS achieved very less allocation of resources (i.e.54%). The execution time of HABCSS was very high when compared with all other optimization techniques. The SFLA-CS achieved the execution of results in 5000ms, whereas the proposed achieved the results in 3800ms. The throughput of DGWO is high (i.e. 75), when compared with other techniques. This shows that the proposed DGWO achieved better results in all parameters which is used to overcome the issues of allocating the resources effectively.

The other parameters such as waiting time and time based evaluation for different task are evaluated in Table 10.

From the Table 7, the proposed method achieved better results in both turnaround time and time based evaluation when compared with existing techniques such as Krill herd and SFLA-CS. When the number of task increases, the turnaround time for proposed method decreases. The time based evaluation is nearly very less for proposed DGWO. For the task 300, the DGWO achieved 15ms in time based evaluation, whereas SFLA-CS and Krill herd achieved $25ms\ and\ 94ms$.

6. Conclusion

The DGWO algorithm is formulated to reduce the knapsack issue for RA mechanism on the CC environment. The DWGO is compared with other existing systems which are executed on the same working platform of JAVA with CloudSim. The execution time for the proposed work is observed as 4000 ms, the throughput is 75 s, and the time required for both turn-around and resource allocation also very less. This research showed that the existing methods such as HABCSS, SFLA-CS, krill herd require high execution time, throughput, and the turn-around time compared to the proposed DWGO method. In future work, efficient energy efficient management can be introduced to reduce the maintenance cost of the system.

Acknowledgments

I thank CHRIST (Deemed To Be University), Bangalore for providing favourable environment to carry out my research work.

References

[1] S.M. Mousavi and F. Gabor, "A novel algorithm for Load Balancing using HBA and ACO in Cloud Computing environment", *International Journal of Computer Science and Information Security*, Vol.14, No.6, pp.48-55, 2016.

[2] M.J. Usman, A.S. Ismail, A.Y. Gital, A. Aliyu, and T. Abubakar, "Energy-Efficient Resource Allocation Technique Using Flower Pollination Algorithm for Cloud Datacenters", In: *Proc. of International Conference of Reliable Information and Communication Technology*, pp.15-29, 2018.

[3] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing", *Future Generation Computer Systems*, Vol.28, No.5, pp.755-768, 2012.

[4] S. Kayalvili and M. Selvam, "Hybrid SFLA-GA algorithm for an optimal resource allocation in cloud", *Cluster Computing*, pp.1-9, 2018.

[5] L. Liu, H. Mei, and B. Xie, "Towards a multi-QoS human-centric cloud computing load balance resource allocation method", *The Journal of Supercomputing*, Vol.72, No.7, pp.2488-2501, 2016.

[6] S.H.H. Madni, S.A.L. Muhammad, and Y. Coulibaly, "Recent advancements in resource allocation techniques for cloud computing environment: a systematic review", *Cluster Computing*, Vol.20, No.3, pp.2489-2533, 2017.

[7] H. Ziafat and S.M. Babamir, "A hierarchical structure for optimal resource allocation in geographically distributed clouds", *Future Generation Computer Systems*, Vol.90, pp.539-568.

[8] A. Singh, D. Juneja, and M. Malhotra, "A novel agent based autonomous and service composition framework for cost optimization of resource provisioning in cloud computing", *Journal of King Saud University-Computer and Information Sciences*, Vol.29, No.1, pp.19-28, 2017.

[9] Z. Zhong, K. Chen, X. Zhai, and S. Zhou, "Virtual machine-based task scheduling algorithm in a cloud computing environment", *Tsinghua Science and Technology*, Vol.210, No.6, pp.660-667, 2016.

[10] K. P. N. Jayasena, L. Li, and Q. Xie, "Multi-modal multimedia big data analyzing architecture and resource allocation on cloud platform", *Neurocomputing*, Vol.253, pp.135-143, 2017.

[11] F. Koch, M. D. Assunção, C. Cardonha, and M. A Netto, "Optimising resource costs of cloud computing for education", *Future Generation Computer Systems*, Vol.55, pp.473-479, 2016.

[12] B. Dhinesh and K. Venkata, "Honey bee behavior inspired load balancing of tasks in cloud computing environments", *Applied Soft Computing*, Vol.13, No.5, pp.2292-2303, 2013.

[13] A. Mosavi, "The large scale system of multiple criteria decision making", *IFAC Proceedings*, Vol.43, No.8, pp.354-359, 2010.

[14] C. Selvaraj, R.S. Kumar, and M. Karnan, "A survey on application of bio-inspired algorithms", *International Journal of Computer Science and Information Technologies*, Vol.5, No.1, pp.366-370, 2014.

[15] A. Mosavi and A. Varkonyi, "Learning in robotics", *International Journal of Computer Applications*, Vol.157, No.1, pp.8-11, 2017.

[16] H. Izakian, B.T. Ladani, A. Abraham, and V. Snasel, "A discrete particle swarm optimization approach for Grid job scheduling", *International Journal of Innovative Computing, Information and Control*, Vol.6, No.9, pp.1-15, 2010.

[17] T. Jena and J. R. Mohanty, "GA-based customer-conscious resource allocation and task scheduling in multi-cloud computing", *Arabian Journal for Science and Engineering*, pp. 1-16, 2017.

[18] X. Liu, X. Zhang, W. Li, and X. Zhang, "Swarm optimization algorithms applied to multi-resource fair allocation in heterogeneous cloud computing systems", *Computing*, Vol.99, No.12, pp.1231-1255, 2017.

[19] H. Zheng, Y. Feng, and J. Tan, "A hybrid energy-aware resource allocation approach in cloud manufacturing environment", *IEEE Access*, Vol.5, pp. 12648-12656, 2017.

[20] M.H. Malekloo, N. Kara, and M. El Barachi, "An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments", *Sustainable Computing: Informatics and Systems*, Vol.17, pp.9-24, 2018.

[21] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, and H. Tenhunen, "Using ant colony system to consolidate vms for green cloud computing", *IEEE Transactions on Services Computing*, Vol.8, No.2, pp.187-198, 2015.

[22] P. Durgadevi and S. Srinivasan, "Resource Allocation in Cloud Computing Using SFLA and Cuckoo Search Hybridization", *International Journal of Parallel Programming*, pp.1-17, 2018.