Cloud Job Scheduling with Ions Motion Optimization Algorithm

Mohammed Elobied Hassan
Faculty of Mathematical Science
University of Khartoum
Khartoum, Sudan
elbashier.mohd@gmail.com

Adil Yousif
College of Sciences and Arts
Najran University
Sharourah, Saudi Arabia
ayalfaki@nu.edu.sa

***Abstract*—Cloud computing technology success comes from its manner of delivering information technology services, how they are designed, propagated, maintained and scaled. Job Scheduling on cloud computing is a crucial research area and is known to be an NP-complete problem. Scheduling refers to assigning user requests to underlying resources effectively. This paper proposes a new Job Scheduling mechanism for cloud computing environment. The proposed mechanism is based on the Ions Motion Optimization (IMO) algorithm. IMO has two phases, liquid, and crystal. These two phases balance the algorithm behavior between convergence and local optima avoidance. To evaluate the proposed mechanism, a simulation with different scenarios using the CloudSim simulator is conducted. The performance of the proposed algorithm is compared with two metaheuristic algorithms known as Cat Swarm Optimization (CSO) and Glowworm Swarm Optimization (GSO). Furthermore, the proposed IMO mechanism is compared with First Come First Served and random solution. The experimental results demonstrated that the proposed mechanism outperformed both CSO and GSO and produced the shortest execution time in all experimental scenarios.**

Keywords-optimization; ions motion; cloud; job scheduling

I. INTRODUCTION

Cloud computing is a new technology based on transferring computation processes from local desktops to remote providers on the internet. Cloud computing services are extensive and provide on-demand access to a pool of computational resources [1-3]. The consistency and stability of cloud services is based on several features such as the scheduling process of jobs. Scheduling is categorized into three levels, namely job level, resource level, and workflow level. In job scheduling, users submit jobs to cloud providers and job scheduling distributes the jobs submitted by the cloud clients to the provider with suitable resources [4-6]. Cloud job scheduling is a job and resource management process based on several factors to increase the overall cloud performance [7, 8]. Cloud jobs may comprise key in data, processing, accessing software, or storage maintaining process. Cloud providers classify jobs based on the service-level agreement (SLA) and requested services. Each user job is assigned to one of the available cloud resources. Cloud providers execute the submitted jobs, and the results are transmitted back to cloud users [9-12]. Job scheduling in cloud computing is an NP complete problem due to the huge amount of tasks submitted by cloud users to cloud providers. The job scheduling process starts when cloud users submit their jobs to cloud providers. The job scheduler in cloud providers requests the resource information service in order to obtain the status of available resources and their features. Then, the job scheduler assigns the jobs to suitable resources based on the job and resources information requirements. Cloud job scheduler allocates several clients jobs to multiple cloud resources. Job scheduling mechanism tries to allocate cloud resources in an optimal way [3, 13-16]. Current job scheduling methods that use a variety of optimization criteria suffer from several issues. The cloud batch systems prefer turnaround time and throughput as job scheduling criteria. However, interactive cloud systems use response time and fairness [17-19]. Different types of cloud job scheduling based on heuristic, metaheuristic and optimization techniques have been proposed [20-22]. Heuristic cloud job scheduling presents an optimal solution based on knowledge theories for obtaining optimal scheduling solutions [23, 24]. Metaheuristic methods are general methods used for job scheduling based on natural inspired optimization methods such as particle swarm optimization and ant colony optimization [25-27]. Optimization techniques provide good solutions but still not the optimal solution. Therefore, there is a need for new job scheduling methods that optimize cloud turnaround and execution times.

This paper proposes a new job scheduling mechanism for cloud computing environment. The proposed mechanism is based on Ions Motion Optimization (IMO) algorithm. IMO has two phases, liquid and crystal. These two phases balance the algorithm behavior between convergence and local optima avoidance.

II. RELATED WORKS

Glowworm Swarm Optimization (GSO) for solving independent job scheduling and allocation problem of cloud computing resources is presented in [28]. The GSO mechanism considers each glowworm as a job scheduling candidate solution. The mechanism starts with an initial random population containing a number of n solutions. After developing the initial random population, GSO calculates the fitness value for each glowworm. The calculation of the fitness

Corresponding author: Adil Yousif

value starts with dividing each job length by the speed of the resource the job is assigned to. Then, GSO calculates the summation of the division results for each glowworm and finds the maximum fitness value. Authors in [29] proposed Cat Swarm Optimization (CSO) for solving the independent job scheduling problem and job allocation to the resources. Each cat represents a candidate solution. The cat population is divided into two modes, Seeking Mode (SM) and Tracing Mode (TM). A new job scheduling mechanism using Firefly Algorithm to minimize the execution time of jobs is presented in [30]. Firefly Algorithm is a nature-inspired metaheuristic algorithm based on the light attractiveness of fireflies. The mapping process considers each firefly as a job scheduling solution. The firefly with less brightness is attracted and moved towards the brighter one. This process continues for several iterations until the algorithm reaches a specified fitness value [30]. Authors in [31] emphasize on the performance of Infrastructure as a Service (IaaS) based on a meta-scheduling model to accomplish enhanced job scheduling in multiple clouds. They proposed a new inter-cloud task scheduling framework and employed policies to improve the performance of participating clouds. A Modified Particle Swarm Optimization (MPSO) method was introduced for cloud job scheduling in [10]. MPSO has two important parameters in cloud scheduling, namely average length and ratio of successful execution. In [32], a hybrid cloud job scheduling mechanism using Shortest Job First and Priority based Scheduling is introduced to enhance the scheduling process. In [33], a chemical reaction optimization method is presented as an optimization method for cloud job scheduling problem. The method maps the cloud job scheduling problem to chemical reaction behaviors. This method has shorter execution time compared to the Firefly Algorithm and GSO according to the experimentation results. The cloud job scheduling problem is modeled using shark smell optimization problem in [34]. The results of the shark smell job scheduling method outperformed the Firefly Algorithms'. A cuckoo-inspired cloud job scheduling mechanism is proposed in [35].

III. IONS MOTION OPTIMIZATION

IMO algorithm [36] is a metaheuristic algorithm proposed for solving optimization problems. IMO as its name suggests is inspired from ion properties in nature, it mimics the attraction and repulsion forces between ions. IMO is population based, it divides the population into two sets of negative and positive charged ions. These ions move in the search space according to a simple rule (ions with the same charge tend to repel each other while ions with opposite charges attract each other) [37, 38]. Ions in IMO can be in two different phases, liquid phase and crystal phase. In liquid phase, ions move around in the search space more freely. The repulsion forces are ignored in this phase to reassure exploration. The only factor that affects the attraction force is the distance between ions, while force is inversely proportional to the distance between ions [38, 39]. As a result, search agents of IMO eventually converge toward a solution in the search space, and the algorithm enters the crystal phase. Crystal phase implies that ions have converged toward a point in the search space. Though, this point could be a local minimum [40, 41] due to the unknown characteristics of the search space. Therefore, crystal phase aims to solve this entrapment by randomly relocating ions in the search space in respect to the best ions. Nevertheless, the mechanism used for transiting from liquid to crystal phases is to check if the average fitness of the worst ions is equal or smaller than the fitness of the best ions [42, 43]. IMO has been applied to solve different computation problems such as tackling the short-term hydrothermal scheduling problem [40], optimum coverage in wireless sensor networks [44], and optimal robot path planning [42].

IV. THE PROPOSED IMO-BASED SCHEDULING MECHANISM

A. Proposed Mechanism Description

IMO starts by initializing a population of search agents (*Ions*). Each Ion is defined by a number of attributes: $position$, $charge$ and $distance$ from the opposite $BestIon$. $Ion\ position$ represents a feasible scheduling solution and is defined as a vector of integer values $\overrightarrow{Ion}$ of n length (where n is the total number of tasks) and its values fall in the range [0 , $m-1$] (where m is the total number of resources). IMO then divides the population of $Ions$ randomly into two groups ($Anion$ and $Cation$), by assigning a value $\in \{0,1\}$ to the $charge$ attribute of each Ion. After $Ions$ have been assigned a $position$ and a $charge$, IMO evaluates the fitness of each Ion using the objective function $f(x)$ and determines $Best\ Anion$ and $Best\ Cation$. Ions with similar charge repel each other while ions with opposite charge attract each other. The $distance$ for every Ion from the $Best\ Anion$ with opposite charge is calculated using (1):

$$AD_{i,j} = \left|A_{i,j} - CBest_j\right|$$
$$CD_{i,j} = \left|C_{i,j} - ABest_j\right| \quad (1)$$

where $AD_{i,j}$ is the distance from $Anion_{i,j}$ in the $j-th$ job and the $BestCation_j$ in the same job, $CD_{i,j}$ is the distance from $Cation_{i,j}$ in the $j-th$ job and the $BestAnion_j$ in the same job. After calculating the distance, we map the distance to the range [0, 1] by:

$$MappedDist_{A_{i,j}} = \frac{AD_{i,j}}{MaxDist}$$
$$MappedDist_{C_{i,j}} = \frac{CD_{i,j}}{MaxDist} \quad (2)$$

where $MaxDist = numberOfResources - 1$. After that, the force can be calculated using the mapped distance by (3):

$$AF_{i,j} = \frac{1}{1+e^{-0.1/MappedDist_{A_{i,j}}}}$$
$$CF_{i,j} = \frac{1}{1+e^{-0.1/MappedDist_{A_{i,j}}}} \quad (3)$$

where $AF_{i,j}$ and $CF_{i,j}$ are the force for the $i-th$ anion and cation in job j respectively.

Now, the new position of anion and cation are updated as in (4) and (5):

$$A_{i,j} = A_{i,j} + AF_{i,j} \times \left(\boldsymbol{Cbest_j} - A_{i,j}\right) \quad (4)$$
$$C_{i,j} = C_{i,j} + CF_{i,j} \times \left(\boldsymbol{Abest_j} - C_{i,j}\right) \quad (5)$$

where $A_{i,j}$ is the element of the $i-th$ Anion in $j-th$ job

dimension and $C_{i,j}$ is the element of the $i-th$ Cation in $j-th$ job dimension.

B. The Proposed IMO Pseudocode

The pseudocode that describes the proposed IMO cloud job scheduling mechanism follows.

Initialize parameters: PopSize, CbestFit, AbestFit, CworstFit, AworstFit, BestIon
Let Pop be the set of ions 1, 2, 3, … , PopSize
Assign random solution (position) to each ion
Assign a charge (Anion or Cation) to each ion
While (stopping criteria not met) do
For each ions do
Evaluate fitness using $f(x)$
Determine BestIon, CbestFit, AbestFit, CworstFit, AworstFit
Calculate distance form the best ion with opposite charge using:
$AD_{i,j} = |A_{i,j} - CBest_j|$ and $CD_{i,j} = |C_{i,j} - ABest_j|$
Calculate force by $AF_{i,j} = \frac{1}{1+e^{-0.1/AD_{i,j}}}$ and $CF_{i,j} = \frac{1}{1+e^{-0.1/CD_{i,j}}}$
End for
If (CbestFit≥CworstFit/2 and AbestFit≥AworstFit/2) then
Get $\Theta_1$andΘ_2 randomly in interval [-1, 1]
Get α randomly in interval [0, 1]
If (α>0.5) then
$A_i = A_i + \Theta_1 \times (Cbest - 1)$
Else
$A_i = A_i + \Theta_1 \times (Cbest)$
End If
Get α randomly in interval [0, 1]
If (α>0.5) then
$C_i = C_i + \Theta_2 \times (Abest - 1)$
Else
$C_i = C_i + \Theta_2 \times (Abest)$
End If
Get α randomly in interval [0, 1]
If (α>0.05) then
Re-initialize C_i and A_i
End If
End If
End while
Output the position of the best ion

C. The Proposed IMO for Cloud Job Scheduling in Details

Assume we have a set of n jobs to be scheduled among a set of m resources where $n > m$ as in Table I.

$$N = [J_1, J_2, J_3, J_4, \dots, J_n],\ M = [R_1, R_2, R_3, R_4, \dots, R_m]$$

Let $n = 6\ jobs$ and $m = 4\ resources$ as in Table II.

TABLE I. EXAMPLE JOBS LENGTH

Job	J1	J2	J3	J4	J5	J6
Cycle	4	8	10	6	12	3

TABLE II. EXAMPLE RESOURCES SPEED

Resource	r1	r2	r3	r4
Cycles per second	8	4	2	6

The solving procedure using IMO algorithm to find an efficient scheduling solution follows.

1) Parameter Initialization

The algorithm begins by defining the population size as $PopSize$=6 and initializing empty variables for best and worst ions of both types as well as a variable to store the best ion.

2) Assign Random Solutions to Each Ion

A solution in job scheduling is a mapping of jobs to resources where the dimension index represents the job index J_i and the number corresponding indicate the resource R_i

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 2 & 4 & 1 \\ 1 & 3 & 4 & 2 & 3 & 2 \\ 2 & 3 & 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 & 4 & 4 \\ 2 & 3 & 3 & 4 & 1 & 1 \\ 3 & 4 & 1 & 2 & 3 & 2 \end{pmatrix}$$

3) Divide Ions Randomly to Anions A_i and Cations C_i

$$\begin{pmatrix} A_1 \\ A_2 \\ C_1 \\ A_3 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 2 & 4 & 1 \\ 1 & 3 & 4 & 2 & 3 & 2 \\ 2 & 3 & 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 & 4 & 4 \\ 2 & 3 & 3 & 4 & 1 & 1 \\ 3 & 4 & 1 & 2 & 3 & 2 \end{pmatrix}$$

4) Evaluate the Fitness of Each Ion using $\boldsymbol{f(I)}$

$$\boldsymbol{f(I)} = \begin{pmatrix} A_1 \\ A_2 \\ C_1 \\ A_3 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 12 \\ 16 \\ 13 \\ 11 \\ 14 \\ 15 \end{pmatrix}$$

5) Determine $CbestFit,\ AbestFit,\ CworstFit,\ AworstFit,\ BestIon$

Assuming a minimization problem (without loss of generality), we determine the best anion and cation respectively by selecting the lowest fitness values among anions and cations:

$$CbestFit = C_1$$
$$AbestFit = A_3$$
$$CworstFit = C_3$$
$$AworstFit = A_2$$
$$BestIon = A_3$$

6) Calculate Distance for Every Ion from the Best Ion with Opposite Charge

The distance is calculated using Cartesian distance:

$$Distance\ from\ A_1\ to\ CbestFit = |A_{1,j} - Cbest_j| = 7$$
$$Distance\ from\ A_2\ to\ CbestFit = |A_{2,j} - Cbest_j| = 4$$
$$Distance\ from\ A_3\ to\ CbestFit = |A_{3,j} - Cbest_j| = 9$$
$$Distance\ from\ C_1\ to\ AbestFit = |C_{1,j} - Abest_j| = 9$$

$$Distance\ from\ C_2\ to\ AbestFit = \left|C_{2,j} - Abest_j\right| = 12$$
$$Distance\ from\ C_3\ to\ AbestFit = \left|C_{3,j} - Abest_j\right| = 3$$

7) Normalize the Distance between Ions

To normalize the distance to the range [0, 1] we need to calculate the maximum distance between two ions using the following equation:

$$MaxDist = (numberOfResources - 1) \times numberOfJobs$$

Then we can map the distance between every two ions using the equation below:

$$MappedDist = \frac{Distance}{MaxDist}$$

Now we calculate the max distance according to our problem: $MaxDist = 18$. Then, max distance is evaluated to obtain the mapped distance to the range [0, 1]

$$MappedDist(A_1 \text{ to } CbestFit) = 0.38$$
$$MappedDist(A_2 \text{ to } CbestFit) = 0.22$$
$$MappedDist(A_3 \text{ to } CbestFit) = 0.50$$
$$MappedDist(C_1 \text{ to } AbestFit) = 0.50$$
$$MappedDist(C_2 \text{ to } AbestFit) = 0.66$$
$$MappedDist(C_3 \text{ to } AbestFit) = 0.16$$

8) Calculate the Attraction Force between Ions

Note that, IMO algorithm assumes that the only factor for computing the attraction force is the distance between ions. The mathematical model is:

$$AF_{i,j} = \frac{1}{1+e^{-0.1/AD_{i,j}}} \text{ and } CF_{i,j} = \frac{1}{1+e^{-0.1/CD_{i,j}}}$$

where $AF_{i,j}$ and $CF_{i,j}$ is the resultant attraction force of anions and cations respectively, and $AD_{i,j}$ represents the resultant distance of $i-th$ anion from the best cation, and $CD_{i,j}$ represents the resultant distance of $i-th$ cation from the best anion. Now, we calculate the attraction force for each ion using above equations:

$$AF_1 = \frac{1}{1+e^{-0.1/0.38}} = 0.56$$
$$AF_2 = \frac{1}{1+e^{-0.1/0.22}} = 0.61$$
$$AF_3 = \frac{1}{1+e^{-0.1/0.50}} = 0.55$$
$$CF_1 = \frac{1}{1+e^{-0.1/0.50}} = 0.55$$
$$CF_2 = \frac{1}{1+e^{-0.1/0.66}} = 0.53$$
$$CF_3 = \frac{1}{1+e^{-0.1/0.16}} = 0.65$$

9) Update the Positions of Anions and Cations According to the Force Value

The positions are updated according to the following equations:

$$A_{i,j} = A_{i,j} + AF_{i,j} \times (Cbest_j - A_{i,j})$$

$$C_{i,j} = C_{i,j} + CF_{i,j} \times (Abest_j - C_{i,j})$$

Calculation of the new position of A_1:

$$A_{1,j} = \begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} & A_{1,5} & A_{1,6} \\ 2 & 3 & 1 & 2 & 4 & 1 \end{pmatrix}$$
$$A'_{1,1} = A_{1,1} + AF_{1,1} \times (Cbest_1 - A_{1,1}) = 2$$
$$A'_{1,2} = A_{1,2} + AF_{1,2} \times (Cbest_2 - A_{1,2}) = 3$$
$$A'_{1,3} = A_{1,3} + AF_{1,3} \times (Cbest_3 - A_{1,3}) \approx 3$$
$$A'_{1,4} = A_{1,4} + AF_{1,4} \times (Cbest_4 - A_{1,4}) \approx 1$$
$$A'_{1,5} = A_{1,j} + AF_{1,5} \times (Cbest_5 - A_{1,5}) \approx 3$$
$$A'_{1,6} = A_{1,6} + AF_{1,6} \times (Cbest_6 - A_{1,6}) \approx 2$$

The new position of A_1 is as follows:

$$A_1 = (2 \quad 3 \quad 3 \quad 1 \quad 3 \quad 2)$$

Calculation of the new position of A_2:

$$A_{2,j} = \begin{pmatrix} A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} & A_{2,5} & A_{2,6} \\ 1 & 3 & 4 & 2 & 3 & 2 \end{pmatrix}$$
$$A'_{2,1} \approx 2$$
$$A'_{2,2} = 3$$
$$A'_{2,3} = 4$$
$$A'_{2,4} \approx 1$$
$$A'_{2,5} \approx 2$$
$$A'_{2,6} \approx 3$$

The new position of A_2 is as follows:

$$A_2 = (2 \quad 3 \quad 4 \quad 1 \quad 2 \quad 3)$$

10) Entering a Crystal Phase

If (*CbestFit*≥*CworstFit*/2 and *AbestFit*≥*AworstFit*/2) then the algorithm will enter a Crystal Phase. The new position of ions will be updated according to the following pseudocode:

Get α randomly in interval [0, 1]
Θ_1 , Θ_2 Are a random numbers in interval [-1, 1]
If (α>0.5) then
$A_i = A_i + \Theta_1 \times (Cbest - 1)$
Else
$A_i = A_i + \Theta_1 \times (Cbest)$
End If
Get α randomly in interval [0, 1]
If (α > 0.5) then
$C_i = C_i + \Theta_2 \times (Abest - 1)$
Else
$C_i = C_i + \Theta_2 \times (Abest)$
EndIf
Get α randomly in interval [0, 1]
If (α>0.05) then
Re-initialize C_i and A_i randomly

V. EVALUATION AND EXPERIMENTATION

This section covers the experimentation conducted to verify the applicability of the proposed mechanism and to confirm its advantages over other metaheuristic mechanisms. Firstly, the proposed mechanism is compared with First Come First Served (FCFS) scheduling method and with a random solution using four different scenarios. Secondly, a simulation comparison is conducted a between IMO and CSO under two different scenarios. Finally, a comparison between IMO, CSO and GSO is conducted.

A. Experimental Settings

The proposed mechanism is simulated using CloudSim simulation model. The population size of IMO is set to 30 ions and the maximum number of iterations to 100. Each scenario is conducted 10 times and the results are recorded. The processing powers of resources are generated randomly between 10 and 70, while the lengths of tasks are between 40 and 150.

B. IMO, FCFS and Random Solution Experiment Scenarios

This section presents a comparison of the four experiment scenarios between the proposed IMO scheduling algorithm, FCFS algorithm and a randomly generated scheduling solution. As seen in Table III and Figure 1, IMO algorithm performs better than the FCFS algorithm and the random solution. Also, it has the best execution.

TABLE III. IMO FCFS AND RANDOM SOLUTION COMPARISON RESULTS

Scenario	Algorithm	Best fitness
First	FCFS	119.37
	RANDOM	54.92
	IMO	30.596
Second	FCFS	155.1479
	RANDOM	88.283
	IMO	64.2411
Third	FCFS	406.913
	RANDOM	211.07
	IMO	90.405
Forth	FCFS	608.2968
	RANDOM	383.8813
	IMO	232.6167

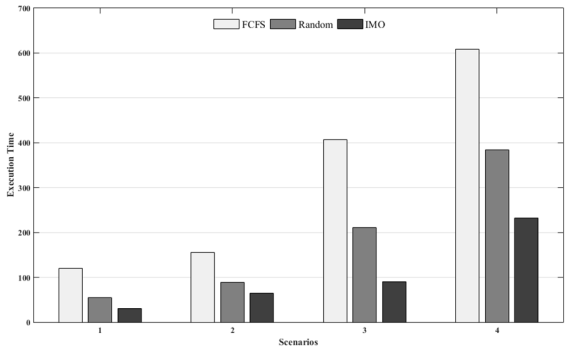

Fig. 1. Comparison between IMO, FCFS and Random Solution

C. IMO and CSO Comparison Result

In this comparison, the processing speed of resources and the length of jobs were generated randomly in the intervals [0, 200] and [0, 400] respectively. Both algorithms used the same jobs and resources specification and both performed 30 runs and the average execution time was calculated.

1) First Scenario

This scenario considers a number of 50 jobs and 20 resources.

TABLE IV. IMO AND CSO 1ST SCENARIO FITNESS (EXECUTION TIME)

Iteration	1	20	40	60	80	100
IMO fitness	181.54	131.50	105.12	80.51	80.51	76.98
CSO fitness	149.93	110.34	107.39	107.39	102.71	100.51

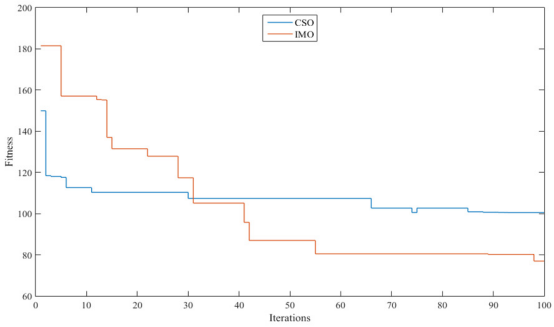

Fig. 2. Comparison between IMO, CSO, first scenario

As shown in Table IV and Figure 2, although CSO starts with a better solution and IMO starts with a slightly high fitness solution, both of them continue to improve the quality of their solutions. CSO maintains its advantage in the first 30 iterations. But as the iterations continue, the IMO found a better solution than CSO in iteration 31, overcame CSO and then continued to improve the performance by reducing execution time.

2) Second Scenario

This scenario considers 100 jobs and 30 resources.

TABLE V. IMO AND CSO 2ND SCENARIO FITNESS (EXECUTION TIME)

Iteration	1	20	40	60	80	100
IMO Fitness	467.79	295.89	242.64	198.78	188.05	162.29
CSO Fitness	492.67	266.16	259.31	259.31	259.31	259.31

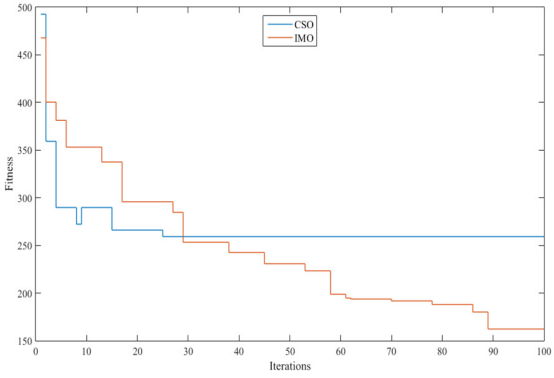

Fig. 3. Comparison between IMO, CSO, second scenario

As shown in Table V and Figure 3, IMO starts with a slightly better solution than CSO, but after the first iteration CSO came up with a better solution. Both algorithms continued to improve their solutions. IMO found a solution better than CSO's after 30 iterations and continued to improve it. CSO remained constant after 30 iterations

D. IMO, CSO and GSO Comparison Result

This section presents a comparison between the proposed mechanism IMO and two other metaheuristic algorithms, CSO and GSO, using 50 jobs and 20 resources generated randomly and used by all the three algorithms.

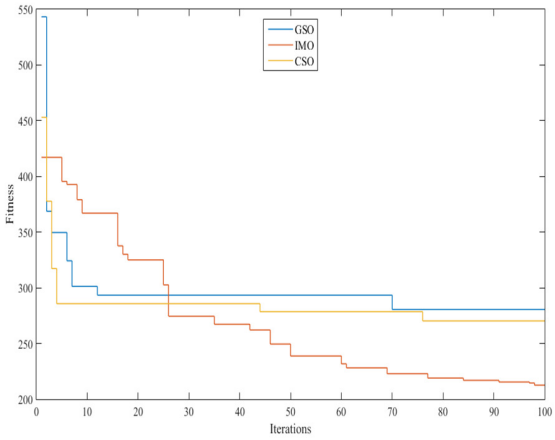

Fig. 4. Comparison between IMO, CSO, and GSO

As shown in Figure 5, IMO started with a good solution but both GSO and CSO were better during the first 20 iterations. IMO found a better solution in less than 30 iterations. Both GSO and CSO did minor improvements to their solutions after 20 iterations and then remained constant while IMO continued to improve its solution until the 96th iteration.

E. Discussion

It was proved that the IMO algorithm can produce promising solutions to scheduling problems. IMO performed 100 iterations in the above scenarios, switching between liquid and crystal phases to overcome local minimum entrapment. Liquid phases were 80% of the total number of iterations, evidencing the exploitation ability of IMO. Crystal iterations were about 20% of the total number of iterations. This proves that the ions do not trap in a local minimum. Moreover, ions in IMO move toward the best ion with opposite charge, thus increasing the number of ions will increase the exploitation of promising regions in the search space. To conclude, IMO starts initializing ions (search agents) randomly, then it changes their location (scheduling solution) and moves them toward the best location found so far. While this will lead to local minimum stagnation, however, crystal phases relocate ions around the best ions found so far randomly.

VI. CONCLUSION

This research proposed a scheduling mechanism for cloud computing jobs based on Ions Motion Optimization. The proposed mechanism considers minimizing the execution time of jobs. This research started by finding an optimal analogy between IMO algorithm and the problem of job scheduling in cloud. IMO begins by initializing randomly a population of ions where each ion position represents a candidate scheduling solution. Then the algorithm divides the population into two groups of anions and cations before evaluating the fitness of each ion using the objective function. The objective function calculates the execution time of each candidate solution. Furthermore, ions move toward the best ions from the other group in the population. IMO continually transits between liquid and crystal phases to maintain a ratio between exploitation and exploration of the search space. Liquid phase guarantees convergence of ions toward the best found solutions and also guarantees exploiting the promising locations found so far. Crystal phase relocates ions randomly around the best solutions found at this point. When compared with two metaheuristic algorithms using the same set of jobs and resources, IMO outperformed them. IMO starts the search with a solution that is slightly better or worse than GSO and CSO, but it overcomes both algorithms after about 30 iterations in all scenarios. IMO also proved its advantage over traditional methods and produced highly competitive solutions having the less execution time when compared to FCFS.

REFERENCES

[1] B. K. Rani, B. P. Rani, A. V. Babu, "Cloud computing and inter-clouds–types, topologies and research issues", Procedia Computer Science, Vol. 50, pp. 24-29, 2015

[2] T. Erl, R. Puttini, Z. Mahmood, Cloud computing: Concepts, technology and architecture, Prentice Hall, 2013

[3] T. Mathew, K. C. Sekaran, J. Jose, "Study and analysis of various task scheduling algorithms in the cloud computing environment", International Conference on Advances in Computing, Communications and Informatics, New Delhi, India, September 24-27, 2014

[4] P. Mell, T. Grance, The NIST definition of cloud computing, National Institute of Standards and Technology, 2011

[5] A. T. Velte, T. J. Velte, R. Elsenpeter, Cloud computing: A practical approach, McGraw-Hill, 2009

[6] B. Furht, Cloud computing fundamentals, Springer, 2010

[7] S. F. Issawi, A. A. Halees, M. Radi, "An efficient adaptive load balancing algorithm for cloud computing under Bursty workloads", Engineering, Technology & Applied Science Research, Vol. 5, No. 3, pp. 795-800, 2015

[8] A. Khattara, W. R. C. Khettaf, M. Mostefai, "An efficient metaheuristic approach for the multi-period technician routing and scheduling problem", Engineering, Technology & Applied Science Research, Vol. 9, No. 5, pp. 4718-4723, 2019

[9] A. R. Arunarani, D. Manjula, V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey", Future Generation Computer Systems, Vol. 91, pp. 407-415, 2019

[10] B. Jana, M. Chakraborty, T. Mandal, "A task scheduling technique based on particle swarm optimization algorithm in cloud environment", in: Soft Computing: Theories and Applications, Proceedings of SoCTA 2017, pp. 525-536, Springer, 2018

[11] M. Haque, R. Islam, M. R. Kabir, F. N. Nur, N. N. Moon, "A priority-based process scheduling algorithm in cloud computing", in: Emerging Technologies in Data Mining and Information Security, Advances in Intelligent Systems and Computing, Vol. 755, pp. 239-248, Springer, 2018

[12] R. Somula, S. Nalluri, M. NallaKaruppan, S. Ashok, G. Kannayaram, "Analysis of CPU scheduling algorithms for cloud computing", in: Smart Intelligent Computing and Applications, Smart Innovation, Systems and Technologies, Vol 105, pp. 375-382, Springer, 2018

[13] O. J. Shirazi, G. Dastghaibyfard, M. M. Raja, "Task scheduling with firefly algorithm in cloud computing", Science International, Vol. 27, No. 1, pp. 167-172, 2014

[14] Y. Miao, "Resource scheduling simulation design of firefly algorithm based on chaos optimization in cloud computing", International Journal of Grid Distributed Computing, Vol. 7, No. 6, pp. 221-228, 2014

[15] M. Aboalama, A. Yousif, "Enhanced job scheduling algorithm for cloud computing using shortest remaining job first", International Journal of Computer Science & Management Studies, Vol. 15, No. 6, pp. 65-68, 2015

[16] Y. P. Dave, A. S. Shelat, D. S. Patel, R. H. Jhaveri, "Various job scheduling algorithms in cloud computing: A survey", International Conference on Information Communication and Embedded Systems, Chennai, India, February 27-28, 2014

[17] D. Oliveira, A. Brinkmann, N. Rosa, P. Maciel, "Performability evaluation and optimization of workflow applications in cloud environments", Journal of Grid Computing, Vol. 17, pp. 749-770, 2019

[18] L. Zhou, L. Zhang, L. Ren, J. Wang, "Real-time scheduling of cloud manufacturing services based on dynamic data-driven simulation", IEEE Transactions on Industrial Informatics, Vol. 15, No. 9, pp. 5042-5051, 2019

[19] L. Mei, W. K. Chan, T. H. Tse, "A tale of clouds: Paradigm comparisons and some thoughts on research issues", IEEE Asia-Pacific Services Computing Conference, Yilan, Taiwan, December 9-12, 2008

[20] S. Mohanty, S. C. Moharana, H. Das, S. C. Satpathy, "QoS aware group-based workload scheduling in cloud environment", in: Data Engineering and Communication Technology: Proceedings of 3rd ICDECT-2K19, pp. 953-960, Springer, 2020

[21] C. Li, C. Wang, Y. Luo, "An efficient scheduling optimization strategy for improving consistency maintenance in edge cloud environment", The Journal of Supercomputing, available at: https://doi.org/10.1007/s11227-019-03133-9, 2020

[22] Z. Tong, H. Chen, X. Deng, K. Li, K. Li, "A scheduling scheme in the cloud computing environment using deep Q-learning", Information Sciences, Vol. 512, pp. 1170-1191, 2020

[23] B. Nayak, S. K. Padhi, P. K. Pattnaik, "Optimization of cloud datacenter using heuristic strategic approach", in: Soft Computing and Signal Processing: Proceedings of ICSCSP 2018, Vol. 1, pp. 91-100, Springer, 2019

[24] S. Ijaz, E. U. Munir, "MOPT: List-based heuristic for scheduling workflows in cloud environment", The Journal of Supercomputing, Vol. 75, pp. 3740-3768, 2019

[25] R. Singh, "Hybrid metaheuristic based scheduling with job duplication for cloud data centers", in: Harmony Search and Nature Inspired Optimization Algorithms: Theory and Applications, ICHSA 2018, pp. 989-997, Springer, 2018

[26] M. Aruna, D. Bhanu, S. Karthik, "An improved load balanced metaheuristic scheduling in cloud", Cluster Computing, Vol. 22, pp. 10873-10881, 2019

[27] H. Singh, S. Tyagi, P. Kumar, "Scheduling in cloud computing environment using metaheuristic techniques: A survey", in: Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018, pp. 753-763, Springer, 2019

[28] D. I. Esa, A. Yousif, "Glowworm swarm optimization (GSO) for cloud jobs scheduling", International Journal of Advanced Science and Technology, Vol. 96, pp. 71-82, 2016

[29] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, A. Al-Khasawneh, "Hybrid cat swarm optimization and simulated annealing for dynamic task scheduling on cloud computing environment", Journal of Information and Communication Technology, Vol. 17, No. 3, pp. 435-467, 2018

[30] D. I. Esa, A. Yousif, "Scheduling jobs on cloud computing using firefly algorithm", International Journal of Grid and Distributed Computing, Vol. 9, No. 7, pp. 149-158, 2016

[31] S. Sotiriadis, N. Bessis, A. Anjum, R. Buyya, "An Inter-Cloud Meta-Scheduling (ICMS) simulation framework: Architecture and evaluation", IEEE Transactions on Services Computing, Vol. 11, No. 1, pp. 5-19, 2018

[32] A. V. Krishna, S. Ramasubbareddy, K. Govinda, "Task scheduling based on hybrid algorithm for cloud computing", International Conference on Intelligent Computing and Smart Communication, Tehri, India, April 20-21, 2019

[33] A. M. Zain, A. Yousif, "Chemical Reaction Optimization (CRO) for cloud job scheduling", SN Applied Sciences, Vol. 2, Article ID 53, 2020

[34] Y. M. Suliman, A. Yousif, M. B. Bashir, "Shark smell optimization (SSO) algorithm for cloud jobs scheduling", International Conference on Computing, Riyadh, Saudi Arabia, December 10-12, 2019

[35] E. Aloboud, H. Kurdi, "Cuckoo-inspired job scheduling algorithm for cloud computing", Procedia Computer Science, Vol. 151, pp. 1078-1083, 2019

[36] B. Javidy, A. Hatamlou, S. Mirjalili, "Ions motion algorithm for solving optimization problems", Applied Soft Computing, Vol. 32, pp. 72-79, 2015

[37] T. T. Nguyen, M. J. Wang, J. S. Pan, T. K. Dao, T. G. Ngo, "A load economic dispatch based on ion motion optimization algorithm", in: Advances in Intelligent Information Hiding and Multimedia Signal Processing, pp. 115-125, Springer, 2019

[38] C. H. Yang, K. C. Wu, Y. S. Lin, L. Y. Chuang, H. W. Chang, "Protein folding prediction in the HP model using ions motion optimization with a greedy algorithm", BioData Mining, Vol. 11, Article ID 17, 2018

[39] M. Kumar, J. S. Dhillon, "An experimental study of ion motion optimization for constraint economic load dispatch problem", International Conference on Power Energy, Environment and Intelligent Control, Greater Noida, India, April 13-14, 2018

[40] S. Das, A. Bhattacharya, A. K. Chakraborty, "Quasi-reflected ions motion optimization algorithm for short-term hydrothermal scheduling", Neural Computing and Applications, Vol. 29, pp. 123-149, 2018

[41] G. Kong, Y. Zhang, A. J. M. Khalaf, S. Panahi, I. Hussain, "Parameter estimation in a new chaotic memristive system using ions motion optimization", The European Physical Journal Special Topics, Vol. 228, pp. 2133-2145, 2019

[42] J. S. Pan, T. T. Nguyen, S. C. Chu, T. K. Dao, T. G. Ngo, "A multi-objective ions motion optimization for robot path planning", International Conference on Engineering Research and Applications, Thai Nguyen, Vietnam, December 1-2, 2019

[43] B. Wang, C. Wang, L. Wang, N. Xie, W. Wei, "Recognition of sEMG hand actions based on cloud adaptive quantum chaos ions motion algorithm optimized SVM", Journal of Mechanics in Medicine and Biology, Vol. 19, No. 6, Article ID 1950047, 2019

[44] T. T. Nguyen, J. S. Pan, T. Y. Wu, T. K. Dao, T. D. Nguyen, "Node coverage optimization strategy based on ions motion optimization", Journal of Network Intelligence, Vol. 4, No. 1, pp. 1-9, 2019