Research Article

Chemical reaction optimization (CRO) for cloud job scheduling

Alneel Mohammed Zain[1] · Adil Yousif[2]

Received: 28 September 2019 / Accepted: 21 November 2019 / Published online: 9 December 2019

Abstract
Cloud computing is an emerging technology that provides functions of traditional computing as services via the Internet. Cloud job scheduling mechanisms try to allocate cloud resources to users' submitted jobs in an optimal way. Several metaheuristic algorithms are used to obtain the optimum scheduling solution, such as genetic algorithm, glowworm swarm optimization and cat swarm optimization. This study introduced an optimal metaheuristic job scheduling method using chemical reaction optimization (CRO). Chemical reaction optimization is a new metaheuristic algorithm inspired from the interactions of molecules to achieve the lowest energy state possible during a chemical reaction. CRO mimics molecules' interaction in chemical reaction microscopically. The CRO mechanism simulates the interactions of chemical reaction molecules to find out the lowest energy value possible. The proposed mechanism represents the molecular structure of each molecule as a vector of integer values, each of which represents a feasible cloud job scheduling solution. The potential energy (*PE*) of each molecule represents its fitness in the objective function that denotes cloud scheduling execution time. Simulation using CloudSim simulator is used to evaluate the proposed CRO mechanism. A comparison with glowworm swarm optimization, cat swarm optimization and first come first served scheduling mechanism is made. The results of simulation showed that the CRO scheduling method has the shortest execution time among all other scheduling mechanisms.

Keywords Chemical reaction optimization · Cloud computing · Scheduling · Metaheuristics

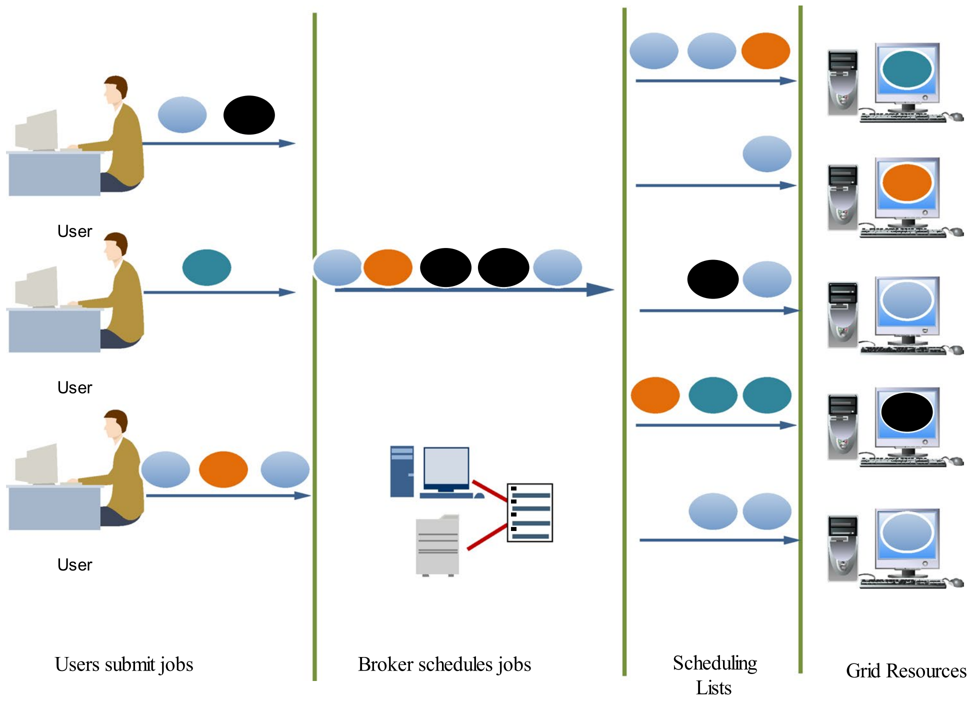
1 Introduction

Computational cloud is an emerged technology based on transferring the computation process from local desktops to remote providers on the Internet. Cloud computing services are extensive and have on-demand access to a pool of computational resources [1–3]. The consistency and stability of cloud services are based on several features such as the scheduling process of jobs. Cloud job scheduling is categorized into three levels: the job level, the resource level and the workflow level [4, 5]. In cloud job scheduling, users send jobs to cloud providers. Job scheduling distributes the jobs submitted by the clients to the provider suitable resources [6–8]. As illustrated in Fig. 1, in the cloud jobs scheduling, the users send their jobs to the central scheduler or broker. The job scheduler in the provider requests the resource information service for obtaining the state of suitable resources and their features and then assigns the jobs on suitable resources based on the job and resource information requirements. Cloud job scheduler allocates several users' jobs to multiple cloud resources. Job scheduling mechanisms try to allocate cloud resources to users' jobs in an optimal way [3, 9–12]. Cloud resources execute the submitted jobs, and the output is sent back to cloud users [4, 13–15].

The cloud jobs scheduling is a challenging issue due to the huge amount of tasks submitted by cloud users to cloud providers. Different types of cloud job scheduling have been proposed based on heuristic, metaheuristic and optimization techniques. Heuristics cloud job scheduling

✉ Adil Yousif, ayalfaki@nu.edu.sa; Alneel Mohammed Zain, alneel199@gmail.com | [1]Faculty of Mathematical Science, University of Khartoum, Khartoum, Sudan. [2]Faculty of Science and Arts, Najran University, Sharurah, Saudi Arabia.

Fig. 1 Cloud job scheduling



presents an optimal solution based on knowledge theories for obtaining optimal scheduling solutions. Optimization is the selection of the best solution based on some features from the solutions' search space. Simply, optimization methods for cloud jobs scheduling try to minimize the jobs' execution times by systematically mapping jobs to resources based on the optimization algorithm procedures. Metaheuristics methods are general methods used for job scheduling based on nature-inspired optimization methods such as particle swarm optimization and ant colony optimization. Although optimization techniques provide a good solution but still not an optimal solution, there is a need for new job scheduling methods to optimize the cloud execution times.

This study introduced an optimized metaheuristic job scheduling method using chemical reaction optimization (CRO). Chemical reaction optimization is a new metaheuristic algorithm inspired from the interactions of molecules to achieve the lowest energy state possible during a chemical reaction. CRO mimics molecules' structure changes in chemical reaction microscopically. In chemistry, a molecule is made up of atoms. Atoms have different features. CRO summarizes all these characteristics under the term "molecular structure" which corresponds to a single solution. Furthermore, each molecule holds two sorts of energies: The first one is the potential energy (PE), and the second one is the kinetic energy (KE). PE corresponds to solution fitness value in terms of energy which is the objective function value.

This paper is organized as follows. Section 2 describes the related works. Section 3 illustrates how chemical reaction optimization works. Section 4 demonstrates the CRO mechanisms, and Sect. 5 describes the experimentations and discussions. We concluded in Sect. 6.

2 Related works

The research by Esa and Yousif [16] proposed glowworm swarm optimization (GSO) to handle the cloud job scheduling problem and distribution of the cloud tasks on available resources based on glowworm swarm optimization mechanism [16, 17]. The GSO mechanism considers each glowworm as a job scheduling solution. The mechanism starts with an initial random population contains a number of solutions. After developing the initial random population, GSO calculates the fitness value for each glowworm. The calculation of the fitness value starts by computing the local execution times of each resource for jobs assigned to that resource. Then, GSO finds the global execution time for each local execution time for each glowworm and finds the maximum fitness value [16]. In the research by Shohdy et al. [5], the proposed cat swarm optimization (CSO) handles the cloud job scheduling problem and distribution of the cloud tasks on available resources. Each cat represents a candidate solution. The cat's population is divided into two modes, seeking mode (SM) and tracing mode (TM).

Esa and Yousif [18] proposed a firefly algorithm metaheuristics job scheduling to optimize the fitness of cloud scheduling solutions. Firefly algorithm is a nature-inspired metaheuristics based on the light attractiveness of fireflies [19, 20]. The mapping process considers each firefly as a job scheduling solution. The firefly that has less brightness is attracted and moved toward the brighter one [10, 21]. This process continues for several iterations until the algorithm reaches a specified fitness value [18].

The researchers in Wang et al. [22] emphasize the efficiency in multiple clouds using meta-scheduling model to accomplish an enhanced job scheduling in multiple clouds. The study proposed a new inter-cloud task scheduling method and employed protocols and rules to improve the efficiency of clouds [22].

Jana et al. [4] presented a modified particle swarm optimization (MPSO) method that has two important factors in scheduling average times and percentage of successful solutions.

3 Chemical reaction optimization (CRO)

Chemical reaction optimization firstly proposed as a new metaheuristic algorithm inspired from the interactions of molecules to achieve the lowest energy state possible during a reaction. CRO mimics what happens to molecules in chemical reaction microscopically. Each reacting system tries to find a minimum of free energy, and hence all chemical reactions try to release energy. Lower energy tends to give more stable molecule [23, 24].

In chemistry, a molecule is made up of atoms. Atoms have different features. CRO summarizes all these characteristics under the term "molecular structure" which corresponds to a single solution. Furthermore, each molecule holds two sorts of energies: The first one is the potential energy (PE), and the second one is the kinetic energy (KE). PE corresponds to solution fitness value in terms of energy which is the objective function value. KE quantifies the forgiveness of tolerating a less stable structure. CRO also employs the conservation of energy law which states that "energy can neither be created nor be destroyed." However, energy is transmitted between molecules or within a molecule. A central buffer is also used to sustain KE of the molecules. Therefore, the molecules tend to have less KE as the algorithm progresses [24, 25].

When molecules collide with each other or with the wall of the container, each collision results in one of the four elementary reactions which are: on-wall ineffective collision, inter-molecular ineffective collision, decomposition and synthesis. These four reactions can be categorized based on two different aspects. Based on the extent of change to the molecular structure, on-wall ineffective collision and inter-molecular ineffective collision result in a small amount of change to the structure of the reacting molecules [22, 26]. Conversely, decomposition and synthesis result in different molecules with a new molecular structure. Based on molecularity, on-wall ineffective collision and decomposition are uni-molecular, while inter-molecular ineffective collision and synthesis involve more than one molecule [22, 24].

The basic idea of CRO tries to explore an intelligent way of different regions of the potential energy surface PES. In CRO, exploitation of the surrounding area in PES is carried out by on-wall ineffective collision and inter-molecular ineffective collision. However, when the algorithm failed to converge to the lowest energy molecule or entrapped in local optima, an exploration is performed through decomposition and synthesis.

CRO algorithm includes three stages: initialization, iteration and the final stage. In the initialization stage, a number of variables and control parameters need to be defined and initialized. Note that CRO is a population-based metaheuristic and its population size varies during algorithm iterations due to the behavior of decomposition and synthesis. In the iteration stage, a series of collisions occurred. However, in order to perform a collision, the algorithm first needs to decide whether the collision involves one molecule or more than one. After that, the algorithm examines the decomposition criterion in the case of uni-molecular collision and synthesis criterion in the case of inter-molecular collision. Decomposition is performed if its criterion is satisfied; otherwise, an on-wall ineffective collision is performed. Similarly, the algorithm performs synthesis or otherwise an inter-molecular ineffective collision if it failed to achieve the synthesis criterion. The iteration criteria keep on the repetition process causing molecules to collide until eventually a stopping criterion is satisfied, and the algorithm enters its final stage where it outputs the solution with the lowest energy [26].

4 The proposed CRO-based scheduling mechanism

The following subsections describe the proposed CRO mechanism for job scheduling in cloud computing.

4.1 The proposed CRO mechanism description

The proposed CRO initializes a population of *M* molecules. Each molecule has a number of properties, i.e., molecular structure and potential energy. In the proposed mechanism, *M* represents the molecular structure of each molecule as a vector of integer values, each of which represents a feasible solution. The potential energy (PE) of each molecule represents its fitness in the objective function. Moreover, each molecule has a kinetic energy (KE), which plays a role of a forgiveness threshold that forgives CRO when it accepts a molecule with a higher potential energy. Imagine that a number of molecules exist in a closed container. These molecules travel and collide either on the walls of the container or with other molecules. Four forms of elementary reactions are produced by the collision.

The four elementary reactions that take place between molecules are described in the following subsections.

i. On-wall ineffective collision

In this elementary reaction, a molecule M hits the wall of the container and this causes small changes to the molecular structure of the molecule. Thus, we get a new molecule M' from an existing M by swapping two values of M molecular structure (vector) randomly.

ii. Inter-molecular ineffective collision

In this reaction, two molecules M_1, M_2 hit each other, causing small changes in both of them. And we get two new molecules $M_{1'}, M_{2'}$ from the existing molecules M_1, M_2 by swapping two values from the molecular structure of each molecule M_1, M_2 randomly.

iii. Decomposition

In decomposition, a molecule M hits the wall which results in a significant change to the molecular structure. Thus, the molecule decomposes into two new molecules $M_{1'}, M_{2'}$. The molecular structure of $M_{1'}$ is formed of the left half of the original molecule M and random values, while $M_{2'}$ is formed of the right half of the original molecule and random values.

iv. Synthesis

In synthesis, two molecules M_1, M_2 collide with each other and form a new molecule M'. We get M' by combining the left half of M_1 with the right half of M_2.

4.2 CRO for cloud job scheduling pseudocode

```
1.  Assign parameters value to $PopSize, Molecoll, InitialKE, KELossRate, \alpha, \beta$
        o PopSize is the population size.
        o Molecoll is a random number ∈ [0, 1]
        o InitialKE is the initial KE that all molecule start with.
        o KELossRate is the loss rate of KE after each reaction
        o $\alpha$ is the decomposition threshold value
        o $\beta$ is the synthesis threshold
2.  Let $PopSize$ be the set of molecules $1, 2, 3, \ldots, PopSize$
3.  For each of the molecules do
4.          Assign random solution to the molecular structure
5.          Calculate the $PE$ by $f(w)$
6.          Assign the $KE$ with $InitialKE$
7.  End for
8.  Let the central energy buffer be $buffer$ and assign $buffer = 0$
9.  While the stopping criteria not met do
10.         Get $t$ randomly in the interval $[0, 1]$
11.         If $t > Molecoll$ then
12.                 Select a molecule $M$ from $Pop$ randomly
13.                 If $\alpha_1 \geq \alpha$ then
14.                 $(M_{1'}, M_{2'}, Success)$ = Decompose $(M, buffer)$
15.                 If $Success$ then
16.                 Remove $M$ from $Pop$
17.                 Add $M_{1'}$ and $M_{2'}$ to $Pop$
18.                 End if
19.                 Else
20.                 Ineff_coll_on_wall $(M, buffer)$
21.                 End if
22.         Else
23.                 Select molecules $M_1, M_2$ from $Pop$ randomly
24.                 If $((KE_{M1'}$ or $KE_{M2'}) < \beta)$ then
25.                 $(M', Success)$ = Synthesis $(M_1, M_2)$
26.                 If $Success$ then
27.                 Remove $M_1, M_2$ from $Pop$
28.                 Add $M'$ to $Pop$
29.                 End if
30.                 Else
31.                 Inter_ineff_coll $(M_1, M_2)$
32.                 End if
33.         End if
34.         Check for any new minimum solution
35.         If (no new minimum solution) then
36.                 $\alpha_1 = \alpha_1 + 1$
37.         End if
38. End while
39. Output the overall minimum solution and its function value
```

4.3 The proposed CRO for cloud job scheduling description

This section describes the proposed CRO for cloud job scheduling process, mapping and equations.

Assume we have a set of n jobs that need to be scheduled among a set of m resources where $n > m$

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

Table 1 Example jobs

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

Table 2 Example resources

Resource	Cycle per second
1	8
2	4
3	2
4	6

Let $n = 5 jobs$ as given in Table 1 and $m = 4$ resources as given in Table 2.

CRO steps to find the optimal scheduling problem

1- Parameters Initialization

$PopSize = 5$ $Molecoll = 0.3$ $InitialKE = 400 KELossRate = 0.6$

2- Random solutions Assignments to the molecular structure of each molecule

ω_1	2	3	1	2	4
ω_2	1	3	4	2	3
ω_3	2	3	4	1	2
ω_4	3	4	1	2	4
ω_5	2	3	3	4	1

3- PE molecule Calculation by $f(\omega)$

						PE_{M1}
M_1	2	3	1	2	4	11

						PE_{M2}
M_2	1	3	4	2	3	15

						PE_{M3}
M_3	2	3	4	1	2	11

						PE_{M4}
M_4	3	4	1	2	4	10

						PE_{M5}
M_5	2	3	3	4	1	13

4- KE Assignment with $InitialKE$

						PE_{M1}	KE_{M1}
M_1	2	3	1	2	4	11	400

						PE_{M2}	KE_{M2}
M_2	1	3	4	2	3	15	400

						PE_{M3}	KE_{M3}
M_3	2	3	4	1	2	11	400

						PE_{M4}	KE_{M4}
M_4	3	4	1	2	4	10	400

						PE_{M5}	KE_{M5}
M_5	2	3	3	4	1	13	400

5- Let the central energy buffer be $buffer$ and

Assign $buffer = 0$

6- While the stopping criteria is not met do

Get t randomly, t in the interval $[0, 1]$, $t = 0.6$

Let $\alpha = 3$

- **α** is used as the criteria of **decomposition**, it represent a threshold value for the number of collisions which have occurred without resulting in a lower function value.

7- **If**$t > Molecoll$**then**

8- Select a molecule from Pop randomly.

M_4 is selected for example

						PE_{M4}	KE_{M4}
M_4	3	4	1	2	4	10	400

9- If ($X \geq \alpha$) then

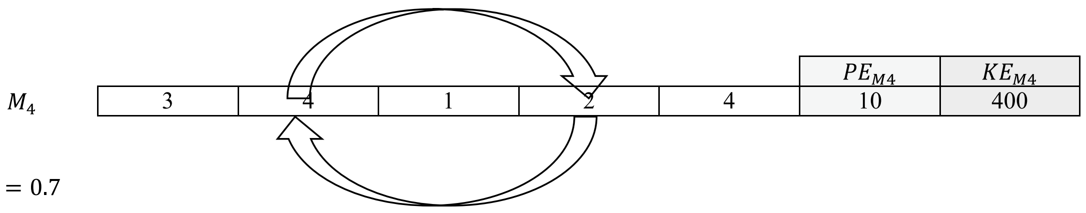
Decompose ($M_4, buffer$)

Else

Ineff_coll_on_wall ($M_4, buffer$)

Ineff_coll_on_wall ($M_4, buffer$)

Two numbers are randomly selected and swapped as shown below.



						PE_{M4}	KE_{M4}
M_4	3	4	1	2	4	10	400

Let $q = 0.7$

$PE_{M4'} = 9$

$$KE_{M4'} = (PE_{M4} + KE_{M4} - PE_{M4'}) \times q = 280$$

$$buffer = buffer + (PE_{M4} + KE_{M4} - PE_{M4'}) \times (1 - q) = 120$$

						$PE_{M4'}$	$KE_{M4'}$
$M_{4'}$	3	2	1	4	4	9	280

The new population

						PE_{M1}	KE_{M1}
M_1	2	3	1	2	4	11	400

						PE_{M2}	KE_{M2}
M_2	1	3	4	2	3	15	400

						PE_{M3}	KE_{M3}
M_3	2	3	4	1	2	11	400

						$PE_{M4'}$	$KE_{M4'}$
M_4	3	2	1	4	4	9	280

						PE_{M5}	KE_{M5}
M_5	2	3	3	4	1	13	400

Decompose (M_4, $buffer$)

						PE_{M4}	KE_{M4}
M_4	3	4	1	2	4	10	400

The selected molecule is decomposed into two new molecules

						PE_{M1}	KE_{M1}
M_1				**2**	**4**		

						PE_{M2}	KE_{M2}
M_2	**3**	**4**	**1**				

Then, the new molecules are completed randomly.

						PE_{M1}	KE_{M1}
M_1	3	1	4	**2**	**4**		

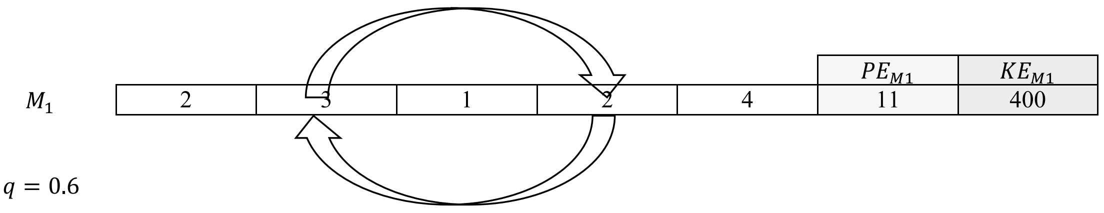
						PE_{M2}	KE_{M2}
M_2	**3**	**4**	**1**	2	1		

10- If ($X < \alpha$) then
Synthesis (M_1, M_2)
Else

Inter_ineff_coll (M_1, M_2)

Inter_ineff_coll $(\boldsymbol{M_1}, \boldsymbol{M_2})$

Two molecules are randomly selected and two numbers are randomly selected from each and swapped as shown below.



						PE_{M1}	KE_{M1}
M_1	2	3	1	2	4	11	400

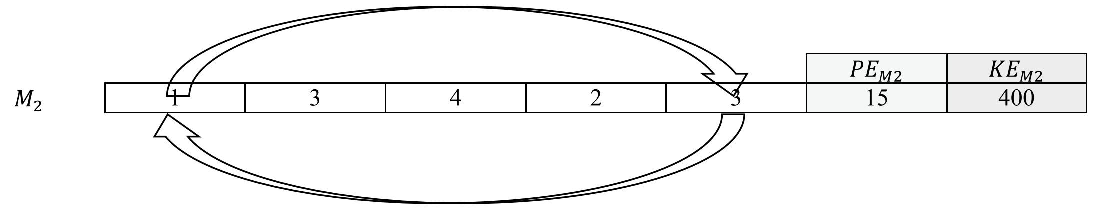
Let $q = 0.6$

$PE_{M1'} = 9$

$$KE_{M1'} = (PE_{M1} + KE_{M1} - PE_{M1'}) \times q = 241$$

$buffer = buffer + (PE_{M1} + KE_{M1} - PE_{M1'}) \times (1 - q) = 160$

						$PE_{M_{1'}}$	$KE_{M_{1'}}$
$M_{1'}$	2	2	1	3	4	9	241



						PE_{M2}	KE_{M2}
M_2	1	3	4	2	3	15	400

$PE_{M2'} = 12$

$$KE_{M2'} = (PE_{M2} + KE_{M2} - PE_{M2'}) \times q = 241$$

$buffer = buffer + (PE_{M2} + KE_{M2} - PE_{M2'}) \times (1 - q) = 161$

						$PE_{M_{2'}}$	$KE_{M_{2'}}$
$M_{2'}$	3	3	4	2	1	12	241

Synthesis $(\boldsymbol{M_1}, \boldsymbol{M_2})$

In synthesis, we select two molecules randomly and combine them into a new one.

						PE_{M1}	KE_{M1}
M_1	2	3	1	**2**	**4**	11	400

						PE_{M2}	KE_{M2}
M_2	**1**	**3**	**4**	2	3	15	400

The new molecule is formed form the left side of the first molecule and the right side of the second molecule.

						$PE_{M'}$	$KE_{M'}$
M'	**1**	**3**	**4**	**2**	**4**	11	400

5 Evaluation and experimentation

This section describes the experiments carried out to assess the proposed chemical reaction optimization mechanism. The first part of the section illustrates the parameter configuration employed for the simulation including the servers utilized to execute the simulation and the cloud infrastructure. The second part describes the benchmark and the parameters for the proposed chemical reaction optimization.

5.1 Experimental settings and system configurations

To carry out the experiments, the study applied a prototype of the proposed chemical reaction scheduling mechanism using CloudSim simulator. The CloudSim software was installed in a DELL server with Intel i7-2760QM CPU 2.70 GHz with 8 cores and memory size of 8 GBs. The evaluation process employed several benchmarking software programs with the Java programming model. The experiments extracted the execution time that represents the evaluation process output. The cloud computing model designed for the experiments is a private cloud hosted by a cloud provider involving two forms of computing servers, AMD and Intel. The details of these servers are described in Table 3.

5.2 CRO parameters settings

The evaluation process and CloudSim simulation setup for chemical reaction optimization are configured according to the standard parameter setting for chemical-reaction-inspired metaheuristic for optimization [24], as described in Table 4.

5.3 Simulation scenarios

To evaluate the proposed job scheduling mechanism, three scenarios are considered. The first scenario setup is a comparison between the CRO scheduling method with first come first served and random solution mechanism. The second scenario contains a comparison with the glowworm metaheuristic algorithm using two experiments. In the last section, this study compared the proposed method with the GSO algorithm and the CSO algorithm.

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

Table 3 Cloud infrastructure configurations

Cloud provider	AMD provider	Intel provider
No. of serves	2–32	2–32
Memory	32 GB	32 GB
Storage	800 GB	800 GB
Virtual machines	2	3
Cores	4	4

Table 4 Chemical reaction optimization parameters

Parameter	QAP value	RSPSP value	CAP value
PopSize	25	10	10
KELossRate	0.8	0.5	0.2
MoleColl	0.2	0.2	0.2
InitialKE	1,000,000	10,000	10,000
αa	1300	200	300
βb	10,000	100	10

5.3.1 CRO, FCFS and random solution experiment scenarios

The first experiment conducts a comparison of execution times between CRO and first come first served (FCFS) algorithm. The experiment employed the same jobs and resources for the two mechanisms.

As illustrated in Table 5 and shown in Fig. 2, the proposed CRO obtained shorter execution time than that of FCFS. Random distribution of tasks on resources performed better than the FCFS algorithm. CRO has the best execution times in all scenarios.

Table 5 and Fig. 2 results revealed that the proposed CRO mechanism for cloud job scheduling efficiently

Table 5 Comparison between CRO and FCFS algorithm

Scenario	Algorithm	Execution time (fitness)
First	FCFS	45.8746
	Random solution	29.8778
	CRO	13.8913
Second	FCFS	387.4896
	Random solution	190.1015
	CRO	36.7805
Third	FCFS	256.8352
	Random solution	216.0965
	CRO	62.5568

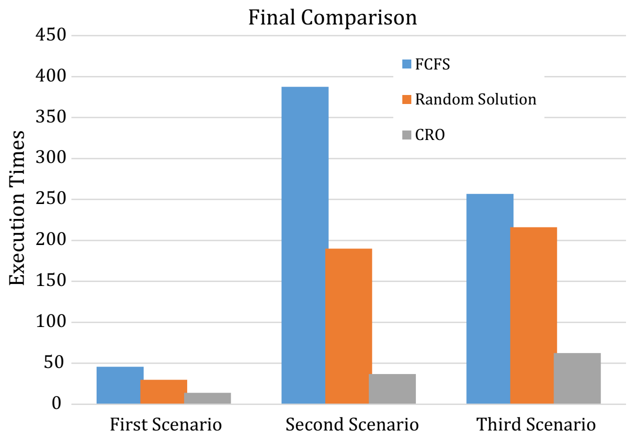


Fig. 2 Comparison results in the last scenario

Table 6 Execution time in the first scenario

Iteration	CRO (fitness)	GSO (fitness)
Initial iteration	139.300259574831	148.283063357991
Last iteration	83.8007445523511	109.934655511343

optimizes scheduling solutions' fitness values. As it can be seen in all scenarios, the execution time decreases as the iterations increase. As shown in Fig. 2, the proposed algorithm has a shorter execution time than that of FCFS and the difference in execution time between CRO and FCFS increases significantly as the number of jobs increases.

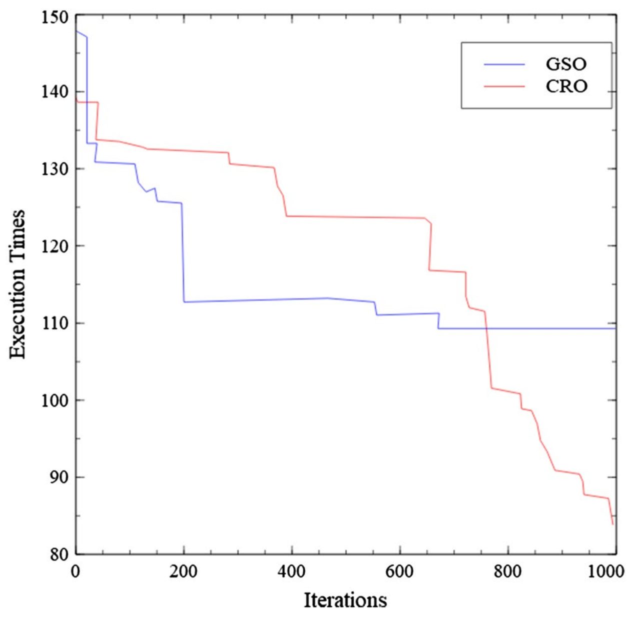


Fig. 3 Comparison results in the first scenario

Therefore, the proposed mechanism is more efficient to work in heavy-load systems than in lightweight systems.

5.3.2 Comparison between CRO and GSO

This section compares the proposed CRO method with the GSO algorithm using two scenarios.

(i) First scenario

In the first scenario, the researchers conducted an experiment with a number of 50 jobs and 20 resources.

As illustrated in Table 6 and shown Fig. 3, CRO starts with a better solution than GSO. In the first few iterations, GSO found better solutions than CRO and continued to improve the solution until iteration 200 when it remained constant and did only a minor improvement. CRO continued to improve its solutions and found better solutions than GSO after 700 iterations and sharply improved its solutions until it ends with a fitness of 83 in the last iteration.

(ii) Second scenario

As shown in Table 7 and Fig. 4, CRO starts with a better solution than GSO and continued to improve the solution in the first 200 iterations. GSO started with a longer solution than CRO; however, it quickly found a better solution in the first few iterations and remained better until iteration 300. CRO found a better solution after 300 iterations and continued to improve the fitness by reducing the execution time.

5.3.3 Comparison between CRO, GSO and CSO

This section presents a comparison between CRO, GSO and CSO.

As shown in Fig. 5, CSO has the shortest execution time compared to CRO and GSO when the experiment started. CRO started with a solution better than GSO and then GSO progressed to find shorter solutions than CRO. Yet again, CRO found a better solution than GSO in iteration number 275; nonetheless, it was not better than CSO. CRO continued to improve its solutions until it reached the shortest solution compared to CSO and GSO after 400 iterations.

5.4 Discussion and technical contributions

The different scenarios of the simulation and analysis proved that the proposed CRO for cloud job scheduling has better execution times than those of cat swarm optimization and first come first served mechanisms. Furthermore, the proposed CRO mechanism is compared with glowworm swarm optimization mechanism and the

Table 7 Execution time in the second scenario

Iteration	CRO (fitness)	GSO (fitness)
Initial iteration	560.053162470686	1008.42773978191
Last iteration	235.012504592497	306.573588842458

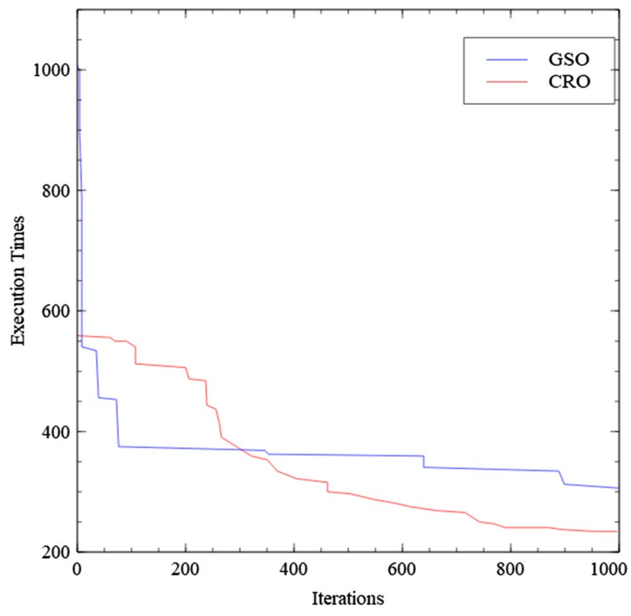


Fig. 4 Comparison results in the first scenario

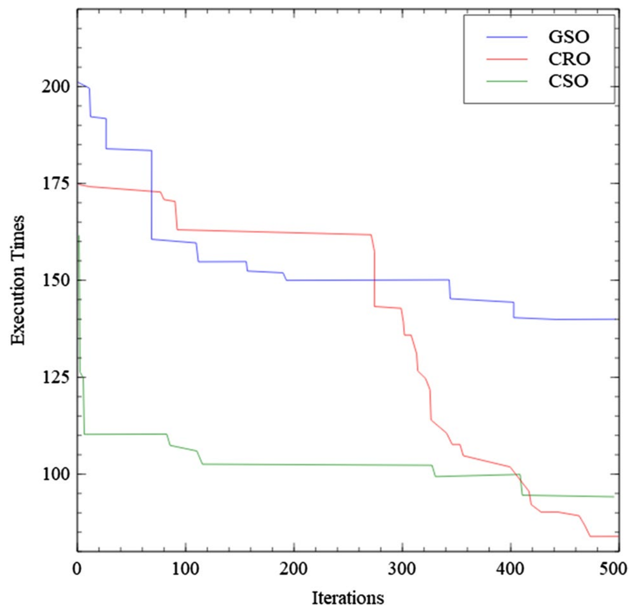


Fig. 5 Comparison results between CRO, GSO and CSO

results revealed the superiority of the proposed CRO over GSO mechanism. The results of simulation of the proposed chemical reaction optimization for cloud job scheduling revealed that the proposed mechanism optimized the cloud jobs' execution times. The reduction in cloud job execution times of the CRO compared to first come first served and cat swarm optimization enhances the efficiency of the cloud system. The expected enhancement of the efficiency of cloud system allows cloud providers to finish user's jobs in a shorter time. Therefore, the proposed CRO mechanism can be integrated with cloud emerging technologies to enhance the performance of cloud computing.

6 Conclusion

This paper proposed a chemical reaction optimization mechanism for cloud jobs scheduling. The proposed mechanism started by initializing a number of molecules that have the same size as the initial scheduling solutions' population size. The structure of molecules is a one-dimensional vector with random values that represents the scheduling solution structure. The execution time is estimated according to the objective function, and the resulting values are the PE of molecules. The initial KE of every molecule is initialized to the value of InitialKE. In each iteration, the study decides whether a uni-molecular or an inter-molecular reaction is applied in the phase according to a comparison result of a random number h [0, 1] with MoleColl. The research determines a suitable subset of molecules to undergo an elementary reaction determined by whether the elementary reaction is uni-molecular or inter-molecular. The iterations continue till a certain condition is fulfilled. Simulation analysis is conducted to evaluate the proposed CRO mechanism. The results revealed that the proposed CRO mechanism has achieved the shortest execution time in the final phase of each scenario. The study compared the proposed mechanism with two metaheuristic algorithms such as GSO and CSO. The experimentation results found out that the proposed CRO outperformed both GSO and CSO in different experiment scenarios. Furthermore, the experiment results evidenced that the performance of the proposed CRO is increased when the workload and the number of iterations increase.

Acknowledgements The authors would like to thank the staff of the computer science department at the University of Khartoum for their valuable comments and advice. Also, we would like to thank Mohammed Elobied Hassan, Dolagi Ezzat and Monika Shohdy for helping in providing comments for the related works part.

Authors' contribution AMZ proposed the use of CRO for cloud job scheduling and developed the CloudSim simulation scenarios. AY described the mapping between cloud job scheduling and CRO metaheuristic.

Availability of data and materials Aneel Mohammed Zain and Adil Yousif confirm that we had full access to all study data, take full responsibility for the accuracy of the data analysis and have authority over manuscript preparation and decisions to submit the manuscript for publication.

Compliance with ethical standards

Conflict of interest The authors declare that they have no competing interests.

References

1. Rani BK, Rani BP, Babu AV (2015) Cloud computing and inter-clouds–types, topologies and research issues. Procedia Comput Sci 50:24–29
2. Erl T, Puttini R, Mahmood Z (2013) Cloud computing: concepts, technology and design. Prentice Hall PTR, Upper Saddle River
3. Mathew T, Sekaran KC, Jose J (2014) Study and analysis of various task scheduling algorithms in the cloud computing environment. In: 2014 International conference on advances in computing, communications and informatics (ICACCI), pp 658–664
4. Jana B, Chakraborty M, Mandal T (2019) A task scheduling technique based on particle swarm optimization algorithm in cloud environment. In: Soft computing: theories and applications. Springer, pp 525–536
5. Shohdy M, Yousif A, Bashir MB (2019) Scheduling jobs on cloud computing using cat swarm optimization (CSO). In: 6th international conference on data science and machine learningapplications (CDMA). KSA
6. Mell P, Grance T (2009) The NIST definition of cloud computing. Natl Inst Stand Technol 53:50
7. Velte T, Velte A, Elsenpeter R (2009) Cloud computing, a practical approach. McGraw-Hill Inc, New York
8. Furht B (2010) Cloud computing fundamentals. In: Handbook of cloud computing. Springer, pp 3–19
9. Jafarzadeh-Shirazi O, Dastghaibyfard G, Raja MM (2015) Task scheduling with firefly algorithm in cloud computing. Sci Int 27:167–171
10. Miao Y (2014) Resource scheduling simulation design of firefly algorithm based on chaos optimization in cloud computing. Int J Grid Distrib Comput 7:221–228
11. Aboalama M, Yousif A (2015) Enhanced job scheduling algorithm for cloud computing using shortest remaining job first (SRJF). Int J Comput Sci Manag Stud 15:65–68
12. Dave YP, Shelat AS, Patel DS, Jhaveri RH (2014) Various job scheduling algorithms in cloud computing: a survey. In: 2014 International conference on information communication and embedded systems (ICICES), pp 1–5
13. Arunarani A, Manjula D, Sugumaran V (2019) Task scheduling techniques in cloud computing: a literature survey. Future Gener Comput Syst 91:407–415
14. Haque M, Islam R, Kabir MR, Nur FN, Moon NN (2019) A priority-based process scheduling algorithm in cloud computing. In: Emerging technologies in data mining and information security. Springer, pp 239–248
15. Somula R, Nalluri S, NallaKaruppan M, Ashok S, Kannayaram G (2019) Analysis of CPU scheduling algorithms for cloud computing. In: Smart intelligent computing and applications. Springer, pp 375–382
16. Esa DI, Yousif A (2016) Glowworm swarm optimization (GSO) for cloud jobs scheduling. Int J Adv Sci Technol 96:71–82
17. Krishnanand K, Ghose D (2010) Glowworm swarm optimization for multimodal search spaces. In: Handbook of swarm intelligence, pp 451–467
18. Esa DI, Yousif A (2016) Scheduling jobs on cloud computing using firefly algorithm. Int J Grid Distrib Comput 9(7):149–158
19. Yousif A, Nor SM, Abdullah AH, Bashir MB (2014) A discrete firefly algorithm for scheduling jobs on computational grid. In: Cuckoo search and firefly algorithm. Springer, pp 271–290
20. Yang XS (2009) Firefly algorithms for multimodal optimization. In: Stochastic algorithms: foundations and applications, pp 169–178
21. Yousif A, Abdullah AH, Nor SM, Bashir MB (2012) Optimizing job scheduling for computational grid based on firefly algorithm. In: 2012 IEEE conference on sustainable utilization and development in engineering and technology (STUDENT), pp 97–101
22. Wang H, Wang W, Zhou X, Sun H, Zhao J, Yu X et al (2017) Firefly algorithm with neighborhood attraction. Inf Sci 382:374–387
23. Pan G, Xu Y, Ouyang A, Zheng G (2015) An improved artificial chemical reaction optimization algorithm for job scheduling problem in grid computing environments. J Comput Theor Nanosci 12:1300–1310
24. Lam A, Li VO (2010) Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evol Comput 14:381–399
25. Cai Z, Niu J, Yang X (2018) A multi measure improved firefly algorithm. In: 2018 2nd IEEE advanced information management, communicates, electronic and automation control conference (IMCEC), pp 20–26
26. Nayak J, Paparao S, Naik B, Seetayya N, Pradeep P, Behera H et al (2019) Chemical reaction optimization: a survey with application and challenges. In: Soft computing in data analytics. Springer, pp 507–524