Volume: 15 Issue: 02 Pages: 5853– 5860 (2023) ISSN: 0975-0290

Task Scheduling Optimization in Cloud Computing by Social Group Optimization Algorithm

Ahmed Y. Hamed
Faculty of Computers and Artificial Intelligence,
Department of Computer Science, Sohag University, Sohag, 82524, Egypt
Email: ayhamedd@gmail.com
M. Kh. Elnahary
Faculty of Computers and Artificial Intelligence,
Department of Computer Science, Sohag University, Sohag, 82524, Egypt
Email: mk409055@gmail.com
Hamdy H. El-Sayed
Faculty of Computers and Artificial Intelligence,
Department of Computer Science, Sohag University, Sohag, 82524, Egypt
Email: hamdy2006x@gmail.com

ABSTRACT

In cloud computing systems, task scheduling is crucial. Task scheduling cannot be done based on a single criterion but rather on rules and regulations that may be referred to as an agreement between cloud customers and providers. This agreement is nothing more than the user's desire for the providers to offer the kind of service that they expect. Providing high-quality services to consumers under the deal is a critical duty for providers, who must also manage many responsibilities. The task scheduling problem may be considered the search for an ideal assignment or mapping of a collection of subtasks of distinct tasks across the available set of resources to meet the intended goals for tasks. This paper proposes an efficient scheduling task algorithm based on the social group optimization of cloud computing systems. By applying it to three cases, we evaluate the performance of our algorithm. The findings suggest that the proposed strategy successfully achieved the best solution in Makespan, Speedup, Efficiency, and Throughput.

Keywords -***Heterogeneous resources, Social Group Optimization Algorithm, Task scheduling, Cloud Computing***

Date of Submission: June 18, 2023 Date of Acceptance: August 06, 2023

1. INTRODUCTION

There is no single description of the cloud, but we may explain it in various ways and techniques. Cloud computing is supercomputing that may be accessed over the internet. It is a shared infrastructure that links big system pools using a variety of ways such as distributed computing, virtualization, and so on. It offers clients a variety of storage, networking, and computing capabilities in the cloud computing environment over the internet, allowing users to store a large quantity of information and access a significant number of processing power using their PCs [1]. The fundamental goal of cloud computing is to manage computing power, storage, numerous platforms, and services assigned to external users on an as-needed basis over the internet. Cloud computing is a rapidly growing computing paradigm that relieves cloud users of the burden of managing hardware, software, networks, and data resources by offloading them to cloud service providers. Clouds provide diverse resources, such as computing platforms, data centres, storage, networks, firewalls, and software. Simultaneously, it provides techniques for regulating these resources, allowing cloud customers to use them without encountering any performance concerns. Cloud Computing Services are grouped into three forms based on the abstraction level and the provider's service model: (1) Infrastructure as a Service (IaaS), (2) Platform as a Service (PaaS), and (3) Software as a Service (SaaS) (SaaS). The key characteristics of cloud computing are distribution, virtualization, and elasticity. Virtualization is a critical component of the cloud. Virtualization is supported by the great majority of software and hardware. We may virtualize and manage diverse components under a cloud platform, including hardware, software, storage, and operating systems [1]. To solve the task scheduling problem satisfactorily, we have presented an efficient method based on a social group optimization algorithm called the efficient social group optimization (ESGO) to decrease the makespan and maximize the Speedup, Efficiency, and Throughput.

The paper is organized as follows: The notations are presented in section 2. Related work is presented in Section 3. problem description is given in Section 4. The social group optimization algorithm is given in Section 5. Section 6 describes the ESGO approach. The evaluation of the proposed algorithm is presented in section 7. Section 8 concludes and offers future work.

2. NOTATIONS

GR	It is the graph of tasks
NS_i	It is the task i
VRM_i	It is the virtual machine i
NVRM	It is the virtual machine's number
NNS	It is the number of tasks
$COM\_COS(NS_i, NS_j)$	It is the communication cost between NS_i and NS_j
$Str\_Time(NS_i, VRM_j)$	It is the start time of task i on a VRM_j
$Fnt\_Time(NS_i, VRM_j)$	It is the finish time of task i on a VRM_j
$Red\_Time(VRM_i)$	It is the V.M.'s ready time i
LIT	It is a list of tasks arranged in topological order of DAG
$Dat\_Arr(TS_i, VM_j)$	It is the time of task's i data arrival to VRM_j

3. RELATED WORK

Cloud computing is a new technology that allows people to pay as they go while still providing outstanding performance. Cloud computing is a heterogeneous system that holds many application data. When scheduling some intense data or computing an intensive application, it is widely understood that minimizing the transferring and processing time is vital to an application programme. The authors create a task scheduling model to lower processing costs and suggest a particle swarm optimization (PSO) approach based on this study's tiny position value rule [2].
Cloud computing has recently overgrown and has established itself as a commercial reality in information technology. Cloud computing is a supplement, consumption, and delivery model for internet-based Information Technology. The scheduling of cloud services impacts the cost-benefit ratio of this computing paradigm provided by service providers to customers. Tasks should be efficiently planned in such a circumstance to reduce execution costs and time. This study [3] proposed a meta-heuristic-based scheduling strategy that minimizes execution time and cost. An improved genetic algorithm is built by integrating two current scheduling algorithms for scheduling tasks while considering their computational cost and processing power.
The efficiency with which infrastructure is constructed and available resources are aggressively used will determine the survival of the next generation of cloud computing. One of the essential concerns in Cloud computing is load balancing, which distributes the dynamic workload over several nodes to ensure that no one resource is overburdened or underutilized. This is an optimization problem, and a skilled load balancer should adapt its approach to the changing environment and task types. The Genetic Algorithm is used in this study [4] to propose a novel load balancing approach (GA).
Scheduling directed acyclic graph (DAG) tasks to reduce makespan has emerged as a significant problem in various heterogeneous computing applications, including task execution order and task-to-processor mapping concerns. The chemical reaction optimization (CRO) technique has lately proven helpful in multiple industries. This paper [5] creates an enhanced hybrid version of the HCRO (hybrid CRO) approach to solve the DAG-based job scheduling issue. In HCRO, the CRO technique is paired with novel heuristic techniques, yielding a new selection strategy. This study provides the following specific contributions. (1) To discover the best local candidate solutions, a Gaussian random walk approach is used. (2) the authors use a left or right rotating shift technique based on maximum Hamming distance to ensure the HCRO algorithm can escape from local optima. (3) A novel selection strategy based on the normal distribution and a pseudo-random shuffling approach is presented to conserve molecular diversity. Furthermore, an exclusive-OR (XOR) operator is put between two strings to reduce the potential of cloning before creating new molecules.
When high efficiency is required, job scheduling is one of the essential considerations in various settings. Different evolutionary strategies have been devised to handle this because task scheduling is a Nondeterministic Polynomial NP-hard problem. Due to the sluggish convergence rate of population-based algorithms, they are paired with local search algorithms. As a result, this work [6] proposes a hybrid particle swarm optimization and hill-climbing strategy to optimize task scheduling timeliness.
This study [7] developed a novel approach dubbed honey bee behaviour inspired load balancing (HBB-LB), which seeks to establish a well-balanced load across virtual machines to maximize throughput. The proposed technique also balances the priority of work on the computers so that the amount of time spent waiting for tasks in the queue is maintained to a minimum.

4. PROBLEM DESCRIPTION

The task scheduling in cloud computing is represented as a Graph with NNS tasks (NS1, NS2, NS3, ..., NSNNS). Each task represents a task with GR and E-directed edges, signifying a portion of the tasks' requests [8]. Each node represents an instruction that might be performed sequentially on the same virtual machine alongside other instructions; it contains one or more inputs. The task an exit or entry task is triggered to execute based on the availability of the inputs. A precedence-constrained partial request result (NSi → NSj), i.e., NSi precedes NSj in the process of execution. The execution time of a task NSi is denoted by (NSi) weight. Let COM_COS(NSi, NSj) be the cost of communication of an edge, and it will be equal to zero if NSi and NSj are scheduled on the same virtual machine. Start and finish times are denoted by Str_Time(NSi, VRMj) and Fnt_Time(NSi, VRMj), respectively [8]. The Dat_Arr of NSi at virtual machine VRMj is given by:

$$Dat\_Arr(NS_i.\ VRM_j) = \max\{Fnt\_Time(NS_k,\ VRM_j) + COM\_COS(NS_i,\ NS_k)\} \quad (1)$$

Where k = 1.2, ..., number of Parents

The task scheduling issue in cloud computing may be characterized as finding the optimal assignment or schedule of the start times of the provided tasks on virtual machines. The scheduled length (completion time) and execution cost are reduced while keeping precedence constrained. The completion time is defined as the schedule length or finish time computed by:

Schedule Length = max(Fnt_Time(NSi, VRMj)) (2)

Fnt_Time(NSi, VRMj) = Str_Time(NSi, VRMj) + WTij(3)

Where i = 1.2. ...., NNS, and j = 1,2, …NVRM

Algorithm 1: To find the schedule length [8]

Input the schedule of tasks
Red_Time[VRM_j] = 0 where j = 1, 2, ……NVRM.
For i = 1 : NNS
{
From LIT take the first task NS_i to be executed and remove it from LIT.
For j = 1 : NVRM
{
If NS_i is scheduled to virtual machine VRM_j
Str_Time(NS_i, VRM_j) = max{Red_Time(VRM_j), Dat_Arr(NS_i, VRM_j)}
Fnt_Time(NS_i, VRM_j) = Str_Time(NS_i, VRM_j) + WT(NS_i, VRM_j)
Red_Time(VRM_j) = Fnt_Time(NS_i, VRM_j)
End If
}
}
Schedule length = max(Fnt_Time)

5. SOCIAL GROUP OPTIMIZATION

In SGO [9], each individual (a potential solution) is endowed with some knowledge and the ability to solve a problem. SGO is a population-based algorithm similar to the other algorithms outlined in the preceding section. For SGO, the population is defined as a group of people (candidate solutions). Everyone gains information and, as a result, has some amount of problem-solving ability. This corresponds to the 'fitness.' The best solution is the most acceptable person. The best individual seeks to spread information among all people, which improves the entire group's knowledge level. The SGO technique is separated into two sections. The first section is the 'improving phase,' while the second part is the 'acquiring phase.' The knowledge level of each member in the group is increased during the 'improving phase,' thanks to the impact of the best person in the group. The best member of the group is the one with the most knowledge and ability to tackle the problem. During the 'acquiring phase,' each individual improves their knowledge by mutual engagement with another member of the group and the best member of the group. The following is a rudimentary mathematical understanding of this notion.Let Z_i, i= 1, 2, 3, . . .N be members of a social group. The social group contains N members, and every member Zi is defined by $Z_i = (Z_{i1}, Z_{i2}, Z_{i3}, \ldots, Z_{iD})$, where D determines the dimensions of a member and Q_i, i= 1, 2, . . .N is their corresponding fitness values, respectively.

Improving phase: The best member (Gbest) in each social group attempts to disseminate information among all individuals, assisting others in the group to increase their knowledge. Hence, G_{best} at generation g is equal min{ Qi , i = 1, 2, . . .N} for solving minimization problem. In the improving phase, each person gets knowledge (here, knowledge refers to the change of traits with the influence of the best person's traits) from the group's best (G_{best}) person. The updating of each person can be computed as follows [9]:

For i = 1 : N
For j=1:D
$Z_{new}(i,j) = e*Z_{old}(i,j)+ran*(G_{best}(j) - Z_{old}(i,j))$ (4)
End for
End for
where ran is a random number, ran ~ U(0, 1)
Accept Z_{new} if it gives a better fitness than Z_{old}
where e is known as self-introspection parameter. Its value can be set from 0 < e < 1.

Acquiring phase: In the acquiring phase, a person of a social group interacts with the best person (G_{best}) and interacts randomly with other persons to acquire knowledge. A person receives new knowledge if the other person has more ability than them. The best knowledgeable person (here known as a person having 'G_{best}') has the most significant influence on others to learn from them. A person will also acquire something new from other persons if they have more knowledge than them in the group. The acquiring phase is expressed as given below [9]:

G_{best} = min{Q (Z_i), i = 1, 2, . . . N} (Z_i's are updated values at the end of the improving phase)
For i = 1 : N
Randomly select one person Z_{ran} , where i ≠ ran
If Q (Z_i) < Q (Z_{ran})
For j = 1 : D
$Z_{new}(i, j) = Z_{old}(i, j) + ran_1 * (Z(i, j) – Z(ran, j) + ran_2 * (G_{best}(j) - Z(i, j))$ (5)
End for
Else
For j = 1 : D
$Z_{new}(i, :) = Z_{old}(i, :) + ran_1 * (Z(ran, :) – Z(i, :) + ran_2 * (G_{best}(j) - Z(i, j))$ (6)
End for
End If
Accept Z_{new} if it gives a better fitness function value.
End for
where ran_1 and ran_2 are two independent random sequences, ran_1 ~ U(0, 1) and ran_2 ~ U(0, 1)

Volume: 15 Issue: 02 Pages: 5853– 5860 (2023) ISSN: 0975-0290

6. THE PROPOSED ALGORITHM

It is clear that the representation of a vector in the social group optimization algorithm is a continuous value form, so we will use the five methods to convert these continuous values to discrete values. The first is the Smallest Position Value (SPV) rule [10], the second is the Largest Position Value (LPV) rule [11], the third is the round nearest function, and the fourth is the floor nearest function, the fifth is Ciel nearest function. In the SPV and LPV, we will use the modulus function with the number of virtual machines and increase the result by one, as shown in Table 1.

Table 1: convert continuous values to discrete values

Population	1.5	2.1	1.3	1.8	3.0	2.5	1.2
SPV rule	7	3	1	4	2	6	5
modulus with SPV and NVRM=3	2	1	2	2	3	1	3
LPV rule	5	6	2	4	1	3	7
modulus with LPV and NVRM=3	3	1	3	2	2	1	2
round nearest function	2	2	1	2	3	3	1
floor nearest function	1	2	1	1	3	2	1
ceil nearest function	2	3	2	2	3	3	2

Algorithm 2: The function that converts a continuous value to a discrete value

```
Function converting(s)
Rando=random number between   [1…5]
If (Rando == 1)
        Use method of SPV rule
Else if (Rando == 2)
        Use method of LPV rule
Else if (Rando == 3)
        Use round nearest function
Else if (Rando == 4)
        Use floor nearest function
Else
        Use ceil nearest function
End if
End function
```

Algorithm 3: ESGO

Input the DAG with communication and computation cost
Initialize the parameters N(number of population), D(dimension), e(self-introspection), uberbound, lowerbound, and maximum iteration
Initialize the population by using population(i,j)=lowerbound + ran*(uberbound-lowerbound)
Convert the initial population by using **Algorithm 2**
Calculate the fitness of each population by using **Algorithm 1**
While iteration <= maximum iteration
Identify the best solution G_{best}
//Improving phase

For i = 1 : N
  For j=1:D
    $Z_{new}(i,j) = e*Z_{old}(i,j)+ran*(G_{best}(j)- Z_{old}(i,j))$
  End for
Convert the new solution by using **Algorithm 2**
Calculate the fitness of the new solution by using **Algorithm 1**
  If (fitness of the new solution < fitness of the old solution Z_i)
    Update the old solution with the new obtained solution
    Update the fitness of the old solution with the new obtained solution
  End for
//Acquiring phase

Identify the best solution G_{best}
For i = 1 : N
  Randomly select one person, Z_{ran} , where i ≠ ran
  If (the fitness of the solution Z_i< the fitness of the solution Z_{ran})
    For j = 1 : D
      $Z_{new}(i, j) = Z_{old}(i, j) + ran_1 * (Z(i, j) – Z(ran, j) + ran_2 * (G_{best} (j) - Z(i, j))$
    End for
  Else
    For j = 1 : D
$Z_{new}(i, :) = Z_{old}(i, :) + ran_1 * (Z(ran, :) – Z(i, :) + ran_2 * (G_{best} (j) - Z(i, j))$
    End for
  End If
  Convert the new solution by using **Algorithm 2**
  Calculate the fitness of the new solution by using **Algorithm 1**
  If (fitness of the new solution < fitness of the old solution Z_i)
    Update the old solution with the new obtained solution
    Update the fitness of the old solution with the new obtained solution
End for
Iteration= iteration+1
End while

7. EVALUATION OF ESGO

We demonstrate the ESGO's performance by applying it to three different instances. The first scenario has eleven tasks and three disparate virtual machines, and the second instance is made up of ten tasks and three different virtual machines. The third is made up of three disparate virtual machines and eleven tasks. We set the Initialize the parameters N(number of population)=100, D(dimension)=number of tasks, e(self-introspection)=0.25, uberbound=3, lowerbound=1, and maximumiteration=100

Volume: 15 Issue: 02 Pages: 5853– 5860 (2023) ISSN: 0975-0290

$$\text{Speedup} = \min_{VRM_{\cdot j}} \left(\sum_{N.S._i} \frac{W.T._{i,j}}{\text{schedule length}} \right) \quad (7)$$

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{NVRM}} \quad (8)$$

$$\text{Throughput} = \frac{\text{NNS}}{\text{Schedule Length}} \quad (9)$$

Case 1: We investigate the scenario of eleven tasks {NS_1, NS_2, NS_3, NS_4, NS_5, NS_6, NS_7, NS_8, NS_9, NS_{10}, NS_{11}} that will be run on three heterogeneous virtual machines {VM_1, VM_2, VM_3}. Table 1 [12] shows the cost of completing each task on different virtual machines. Table 2 shows the start and finish times of each task on different virtual machines and the ESGO schedule. Table 3 shows the comparative results for makespan between ESGO and other algorithms. The ESGO findings are compared to the outcomes of HEFT [12], CPOP [12], and MHEFT [12]. Figures 1, 2, 3, and 4 show the results of the ESGO, HEFT, CPOP, and MHEFT in terms of makespan, speedup, efficiency, and throughput.

Table 1: Computation Cost for Case 1

Task	VM_1	VM_2	VM_3
NS_1	16	19	27
NS_2	18	15	13
NS_3	21	12	22
NS_4	15	13	11
NS_5	22	19	20
NS_6	13	09	11
NS_7	8	11	16
NS_8	14	23	10
NS_9	28	32	12
NS_{10}	15	13	09
NS_{11}	14	16	22

Table 2: Schedule obtained by ESGO for case 1

	VM_1		VM_2		VM_3	
	Str_Time	Fnt_Time	Str_Time	Fnt_Time	Str_Time	Fnt_Time
NS_1	0	16	-	-	-	-
NS_2	-	-	-	-	33	46
NS_3	-	-	36	48		
NS_4	38	53	-	-	-	-
NS_5	16	38	-	-	-	-
NS_6	-	-	72	81		
NS_7	-	-	-	-	57	73
NS_8	-	-	-	-	73	83
NS_9	-	-	-	-	83	95
NS_{10}	-	-	94	107	-	-
NS_{11}	-	-	107	123	-	-

Table 3: the comparative results for case 1

Algorithm	Makespan
CPOP	136
HEFT	134
MHEFT	133
ESGO	123

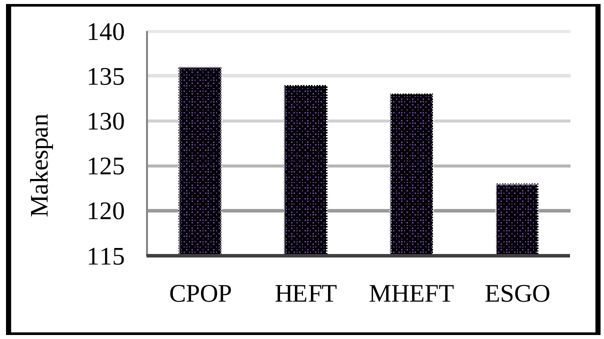


Figure 1: comparison of makespan for case 1

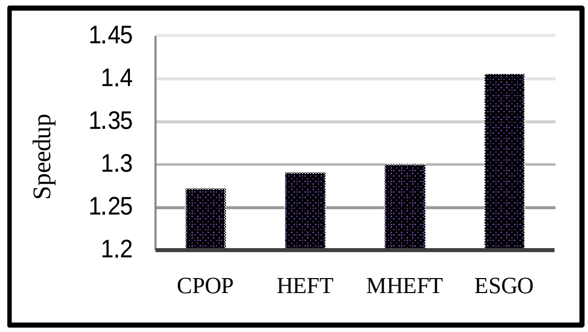


Figure 2: comparison of speedup for case 1

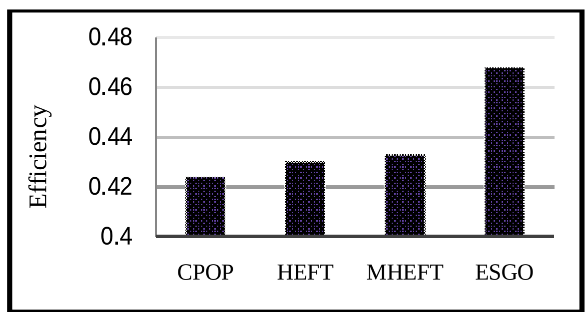


Figure 3: comparison of efficiency for case 1

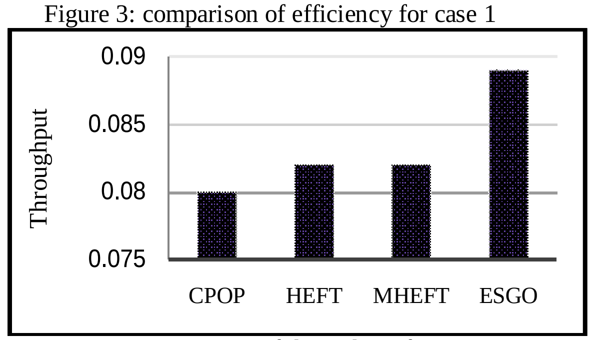


Figure 4: comparison of throughput for case 1

Volume: 15 Issue: 02 Pages: 5853– 5860 (2023) ISSN: 0975-0290

Case 2: We investigate the scene of ten tasks {NS_0, NS_1, NS_2, NS_3, NS_4, NS_5, NS_6, NS_7, NS_8, NS_9} that will be run on three heterogeneous virtual machines {VM_1, VM_2, VM_3}. Table 4 [5] shows the cost of completing each task on different virtual machines. Table 5 shows the start and finish times of each task on different virtual machines and the ESGO schedule. Table 6 shows the comparative results for makespan between ESGO and HCRO [5]. The ESGO findings are compared to the outcomes of HCRO. Figures 5, 6, 7, and 8 show the results of the ESGO and HCRO in terms of makespan, speedup, efficiency, and throughput.

Table 4: Computation Cost for Case *2*

Task	VM_1	VM_2	VM_3
NS_0	10	11	11
NS_1	9	10	8
NS_2	8	6	8
NS_3	10	10	9
NS_4	13	12	13
NS_5	3	2	4
NS_6	10	8	9
NS_7	2	2	2
NS_8	18	17	16
NS_9	15	14	14

Table 5: Schedule obtained by ECS for case 3

	VM_1		VM_2		VM_3	
	Str_Time	Fnt_Time	Str_Time	Fnt_Time	Str_Time	Fnt_Time
NS_0	-	-	-	-	0	11
NS_1	13	22	-	-	-	-
NS_2	-	-	-	-	11	19
NS_3	-	-	-	-	19	28
NS_4	-	-	12	24	-	-
NS_5	-	-	25	27	-	-
NS_6	27	37	-	-	-	-
NS_7	-	-	30	32	-	-
NS_8	-	-	-	-	28	44
NS_9	-	-	-	-	44	58

Table 6: the comparative results for case 2

Algorithm	Makespan
HCRO	61
ESGO	58

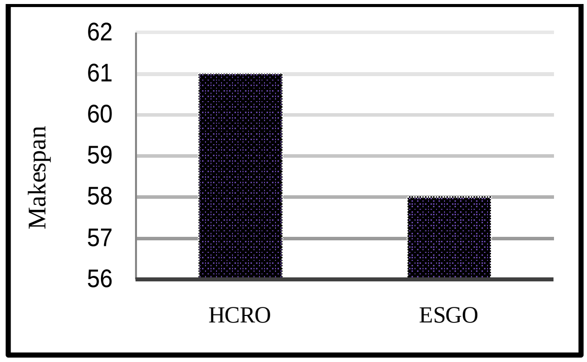


Figure 5: comparison of makespan for case 2

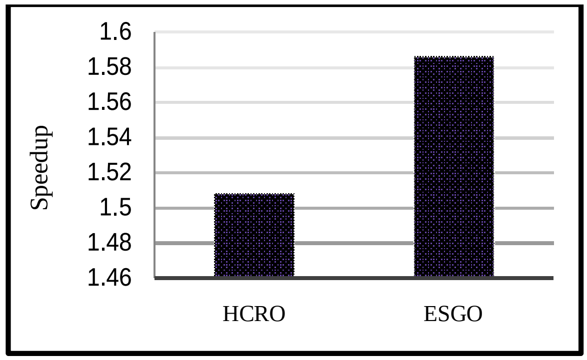


Figure 6: comparison of speedup for case 2

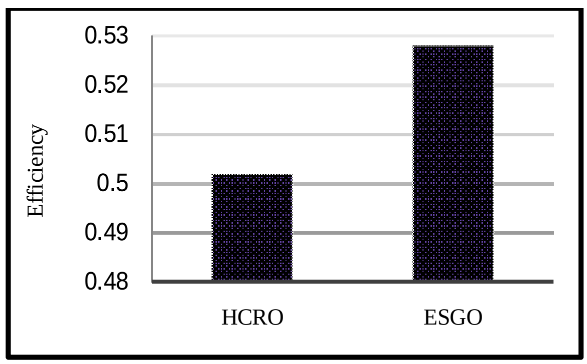


Figure 7: comparison of efficiency for case 2

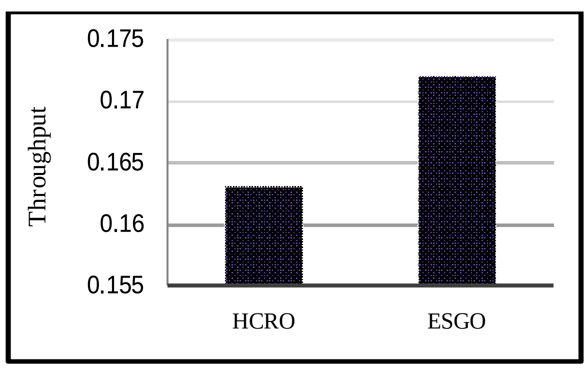


Figure 8: comparison of throughput for case 2

Case 3: We investigate the scenario of eleven tasks {NS_0, NS_1, NS_2, NS_3, NS_4, NS_5, NS_6, NS_7, NS_8, NS_9, NS_{10}} that will be run on three heterogeneous virtual machines

{VM_1, VM_2, VM_3}. Table 7 [13] shows the cost of completing each task on different virtual machines. Table 8 shows the start and finish times of each task on different virtual machines and the ESGO schedule. Table 9 shows the comparative results for makespan between ESGO and other algorithms. The ESGO findings are compared to the outcomes of Upward Rank [14], Downward Rank [14], Level Rank [14], BGA [15], and GA DE HEFT [13]. Figures 9, 10, 11, and 12 show the outcomes of the ESGO, Upward Rank, Downward Rank, Level Rank, BGA, GA_DE_HEFT in terms of makespan, speedup, efficiency, and throughput.

Table 7: Computation Cost for case 3

Task	VM_1	VM_2	VM_3
NS_0	9	11	10
NS_1	11	7	9
NS_2	8	6	4
NS_3	6	5	7
NS_4	9	17	10
NS_5	7	5	9
NS_6	12	15	9
NS_7	17	12	13
NS_8	8	12	10
NS_9	16	15	14
NS_{10}	11	10	12

Table 8. Schedule obtained by ESGO for case 3

	VM_1		VM_2		VM_3	
	Str_Time	Fnt_Time	Str_Time	Fnt_Time	Str_Time	Fnt_Time
NS_0	0	9	-	-	-	-
NS_1	-	-	21	28	-	-
NS_2	-	-	-	-	23	27
NS_3	9	15	-	-	-	-
NS_4	-	-	35	52	-	-
NS_5	-	-	-	-	27	36
NS_6	15	27	-	-	-	-
NS_7	-	-	53	65	-	-
NS_8	43	51	-	-	-	-
NS_9	27	43	-	-	-	-
NS_{10}	-	-	66	76	-	-

Table 9: the comparative results for case 3

Algorithm	Makespan
Upward Rank	88
Downward Rank	87
Level Rank	87
BGA	85
GA_DE_HEFT	78
ESGO	76

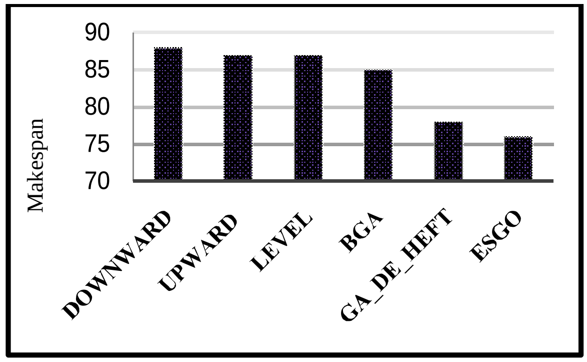


Figure 9: comparison of makespan for case 3

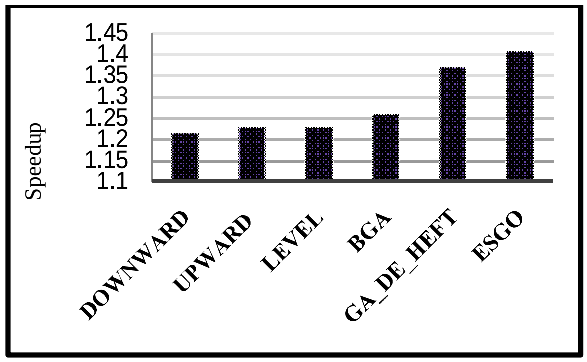


Figure 10: comparison of speedup for case 3

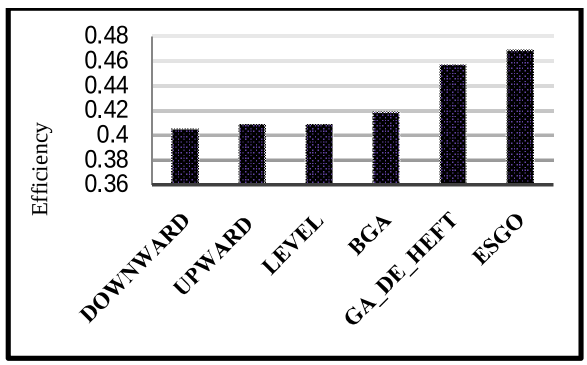


Figure 11: comparison of efficiency for case 3

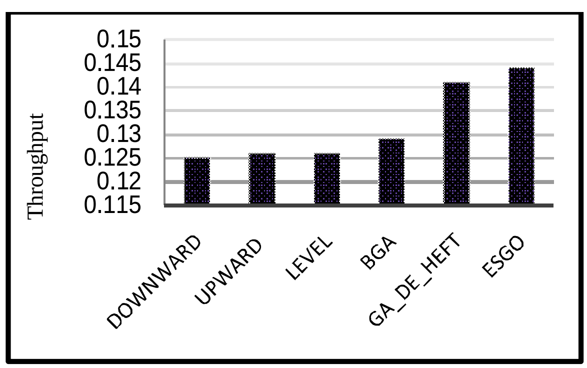


Figure 12: comparison of throughput for case 3

8. CONCLUSION AND FUTURE WORK

The proposed efficient social group optimization algorithms allocate or schedule subtasks to available virtual machines in a cloud computing environment. According to the obtained results on DAGs of different cases, the efficient social group optimization algorithms are significantly more effective than other algorithms in terms of makespan, speedup, efficiency, and throughput. In the future, we will develop an algorithm based on DAGs by considering the load balancing of the resources.

REFERENCES

[1] R.M. Singh, S. Paul, A. Kumar, Task Scheduling in Cloud Computing : Review, 5 (2014) 7940–7944.

[2] L. Guo, S. Zhao, S. Shen, C. Jiang, Task scheduling optimization in cloud computing based on heuristic Algorithm, J. Networks. 7 (2012) 547–553. https://doi.org/10.4304/jnw.7.3.547-553.

[3] S. Kaur, A. Verma, An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment, Int. J. Inf. Technol. Comput. Sci. 4 (2012) 74–79. https://doi.org/10.5815/ijitcs.2012.10.09.

[4] K. Dasgupta, B. Mandal, P. Dutta, J.K. Mandal, S. Dam, A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing, Procedia Technol. 10 (2013) 340–347. https://doi.org/10.1016/j.protcy.2013.12.369.

[5] Y. Xu, K. Li, L. He, L. Zhang, K. Li, A Hybrid Chemical Reaction Optimization Scheme for Task Scheduling on Heterogeneous Computing Systems, IEEE Trans. Parallel Distrib. Syst. 26 (2015) 3208–3222. https://doi.org/10.1109/TPDS.2014.2385698.

[6] N. Dordaie, N.J. Navimipour, A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments, ICT Express. 4 (2018) 199–202. https://doi.org/10.1016/j.icte.2017.08.001.

[7] L.D. Dhinesh Babu, P. Venkata Krishna, Honey bee behavior inspired load balancing of tasks in cloud computing environments, Appl. Soft Comput. J. 13 (2013) 2292–2303. https://doi.org/10.1016/j.asoc.2013.01.025.

[8] A.Y. Hamed, M.H. Alkinani, Task scheduling optimization in cloud computing based on genetic algorithms, Comput. Mater. Contin. 69 (2021) 3289–3301. https://doi.org/10.32604/cmc.2021.018658.

[9] S. Satapathy, A. Naik, Social group optimization (SGO): a new population evolutionary optimization technique, Complex Intell. Syst. 2 (2016) 173–203. https://doi.org/10.1007/s40747-016-0022-8.

[10] I. Dubey, M. Gupta, Uniform mutation and SPV rule based optimized PSO algorithm for TSP problem, Proc. 2017 4th Int. Conf. Electron. Commun. Syst. ICECS 2017. 17 (2017) 168–172. https://doi.org/10.1109/ECS.2017.8067862.

[11] L. Wang, Q.K. Pan, M.F. Tasgetiren, A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem, Comput. Ind. Eng. 61 (2011) 76–83. https://doi.org/10.1016/j.cie.2011.02.013.

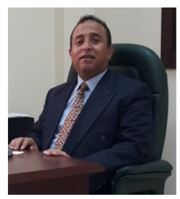
[12] K. Dubey, M. Kumar, S.C. Sharma, Modified HEFT Algorithm for Task Scheduling in Cloud Environment, Procedia Comput. Sci. 125 (2018) 725–732. https://doi.org/10.1016/j.procs.2017.12.093.

[13] A. Kamalinia, A. Ghaffari, Hybrid Task Scheduling Method for Cloud Computing by Genetic and DE Algorithms, Wirel. Pers. Commun. 97 (2017) 6301–6323. https://doi.org/10.1007/s11277-017-4839-2.

[14] H. Topcuoglu, S. Hariri, M.Y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (2002) 260–274. https://doi.org/10.1109/71.993206.

[15] S. Gupta, G. Agarwal, V. Kumar, Task scheduling in multiprocessor system using genetic algorithm, ICMLC 2010 - 2nd Int. Conf. Mach. Learn. Comput. (2010) 267–271. https://doi.org/10.1109/ICMLC.2010.50.

Biographies and Photographs



A. Younes received his PhD degree in Sept. 1996 from South Valley University, Egypt. His research interests include Artificial Intelligence and genetic algorithms; specifically, in the area of computer networks. Recently, he has started conducting a research in the area of Image Processing. Currently, he works as an Professor SohagUniversity, Egypt. Younes always publishes the outcome of his research in international journals and conferences.

M. Kh. Elnahary Received the B.S degree from computer science department, SohagUniversity, Egypt. His interests in task scheduling and computer networks.

Hamdy H. El-Sayed Received the PhD degree in wireless ad hoc network routing protocols from computer science department sohag university Egypt march ,2015. His research interests are in the areas of ad hoc routing protocols and sensor networks, Internet of Things, cloud computing and security.