International Journal of Applied Metaheuristic Computing
Volume 13 • Issue 1

A Deadline-Constrained Time-Cost-Effective Salp Swarm Algorithm for Resource Optimization in Cloud Computing

Richa Jain, Banasthali Vidyapith, India*
Neelam Sharma, Banasthali Vidyapith, India

ABSTRACT

Cloud computing has become the most attractive platform, which provides anything as a service (XaaS). Many applications may be developed and run on the cloud without worrying about platforms. It is a big challenge to allocate optimal resources to these applications and satisfy user quality of service requirements. In this paper, a deadline constrained time-cost effective salp swarm algorithm (DTC-SSA) is proposed to achieve optimized resource allocation. DTC-SSA assigns the user's task to an appropriate virtual machine (Vm) and achieves a trade-off between cost and makespan while satisfying the deadline constraints. Rigorous examination of the algorithm is conducted on the various scale and cloud resources. The proposed algorithm is compared with particle swarm optimization (PSO), grey wolf optimizer(GWO), bat algorithm (BAT), and genetic algorithm (GA). Simulation results prove that it outperforms others by minimizing makespan, execution cost, response time, and improving resource utilization throughput.

KEYWORDS

Cloud Computing, Cost, Deadline, Makespan, Meta-Heuristic Algorithm, Resource Optimization, Task Scheduling, Virtual Machine

1. INTRODUCTION

Cloud computing is becoming an emerging era that provides significant computational assets via network using the pay-as-you-go model during the last decade. It has reshaped all computing models, such as grid computing, parallel computing, and distributed computing, by providing many services through the internet(Panda et al.,2017). Cloud computing has unique features such as on-demand any time self-service, rapid elasticity, and resource pooling pooling (Mell et al, 2011), which draws many people and businesses to hire cloud offerings to run their applications without knowing much about the underlying infrastructure. They can also build their computing platforms on virtual machines provided by the cloud.

In Cloud computing, infrastructure as a Service (IaaS) plays as the foundation block for Software as Service(SaaS) and Platform as Service(PaaS)(Bhardwaj et al.,2010). Amazon EC2, Digital Ocean, Microsoft Azure is the major IaaS providers. These service providers have different categories of the virtual machine, characterized based on configuration, price model, and QoS, from which clients can select as per their requirements. On the other hand, IaaS provider's context, a clients' request for service is periodic, and it is impossible to predict the type and no of VMs' instance. Therefore, it is challenging for a service provider to satisfy clients' requests with its resources while maintaining

DOI: 10.4018/IJAMC.292509 *Corresponding Author

promised QoS. A simple solution of this problem is that the service provider must buy resources in advance, but it is not cost-efficient (Brietland et al.,2011). Another way is Cloud Federation (Toosi et al.,2011), in which the service provider can borrow resources from another service provider. In cloud computing, task Scheduling is a crucial activity responsible for the task and suitable resource (Mao et al.,2015). It needs to find optimized results (Ullman.,1975).

Figure 1 represents the cloud computing model for task scheduling. There are five critical elements in this model (Kumar et al.,2019).

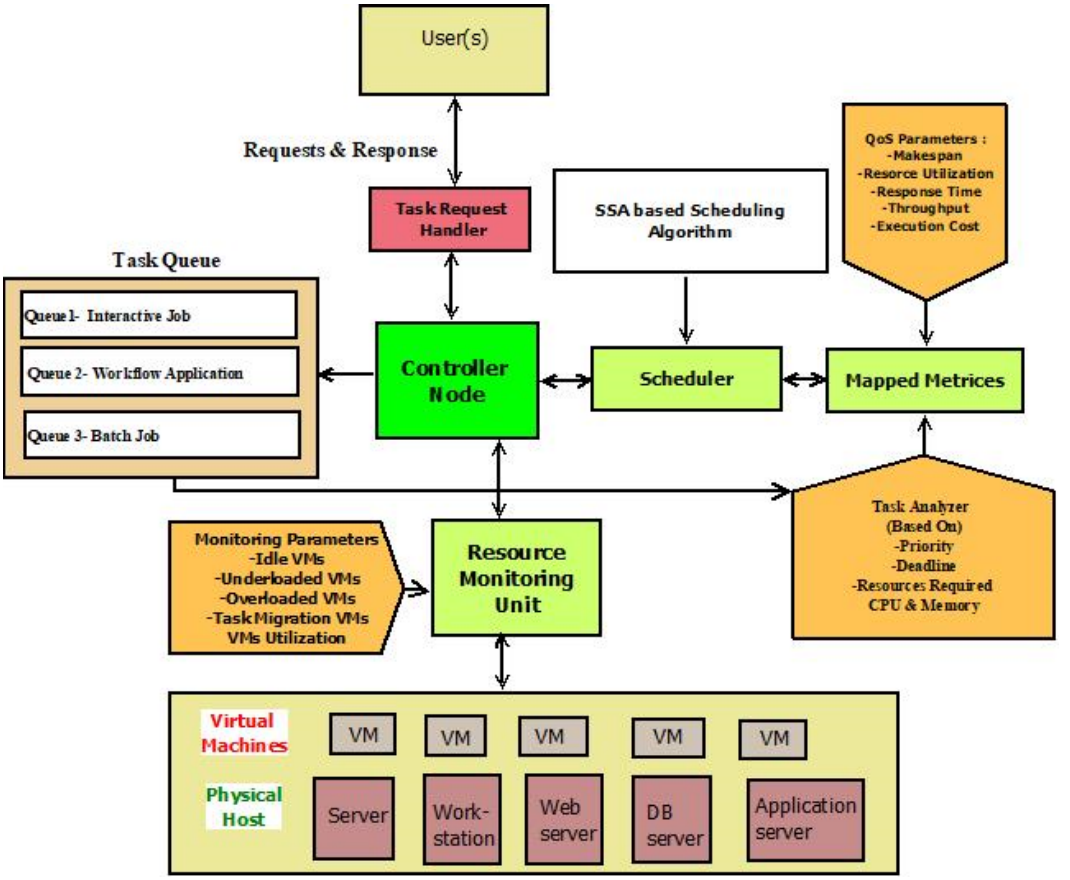
1) Task request handler (TRH) - After submitting the user's task request, TRH handles these requests and decides whether to accept or reject them. It forward selected requests to the controller node.
2) Controller node(resource manager)- Resource manager(RM) handles all incoming and outgoing tasks from other units like TRH, scheduler, and resource monitoring unit as shown in figure 1.RM acts like resource broker. It sends all jobs to the task queue for further processing.
3) Mapped matrices- Tasks are analyzed based on priority, deadline, required CPU power and memory, and the number of resources needed. Then they are submitted to mapped matrices (MM), and so based on this information; MM can find suitable resources and satisfy the user's QoS constraints.
4) Scheduler- Task scheduler maps tasks with the best resource among a pool of available resources and optimizes QoS parameters using a scheduling algorithm. In this paper, Salp Swarm Algorithm (SSA) is used for scheduling.
5) Resource monitoring Unit (RMU): RMU monitors all information related to virtual machines (VMs). After a time interval, it monitors the status of VMs, whether they are over-utilized or under-utilized, and transfers tasks from over-utilized VM to under-utilized VM.

During the allocation of appropriate resources to tasks, mainly four QoS requirements need to be satisfied, i.e., minimum execution time (makespan), minimum execution cost, minimum consumption of energy, and increasing the service provider's profit. Task scheduling belongs to NP-complete, and heuristic algorithms cannot provide global optimum effectively for this. Many works are proposed (Sakellariouet al., 2004,Parsa et al.,2009, Kwok et al.,1996, Maheswaran et al.,1999) based on heuristic approaches. Heuristic algorithms are problem dependent, take "reasonable" computational time and look for "'good enough'" solutions. In contrast, meta-heuristics are higher-level problem-independent frameworks (Yang et al.,2009, Bianchi et al.,2009) that provide solutions for a wide range of problems and provide strategies to develop a lower-level heuristic algorithm. Many researchers are already addressed and proved that metaheuristic algorithms perform better (Laarhoven et al.,1992, Hilliard et al. 1988, Colorni et.al.,1994, Zhang et al.,2005). Day by day, they are getting popular due to their simplicity, accuracy, flexibility, and gradient-free nature. Some nature-inspired metaheuristic techniques are widely explored, e.g., Particle Swarm Optimization (PSO) (Eberhart et al.,1995) shows the social and natural behavior of swarm. The ant colony optimization (ACO)))(Colorni et al.,1991) simulates the ant's behavior for searching the shortest path for food. Others in the literature are Cuckoo Search (CS) algorithm (Yang et al.,2009), Harmony Search(Geem et al.,2001), Artificial Bee Colony (ABC) algorithm (Karaboga et al. 2007), Bat Algorithm (BA)(Yang et al. 2010), Grey Wolf Optimizer (GWO) (Mirjalili et al.,2014).

Despite the excellence of these algorithms, No Free Lunch (NFL)(NFLWolpert et al.,1997)has proved that none of these algorithms can figure out all optimization problems. So there is always a scope for improvement in scheduling. It is the motivation of this proposed work.

Deadline Constrained Time-Cost effective Salp Swarm Algorithm for Task Scheduling in Cloud Computing (DTC-SSA). This algorithm is based on the natural behavior of salp while searching and navigating for food in the sea. It is a swarm-inspired collective algorithm that belongs to a stochastic class. The stochastic algorithm always finds distinct solutions for a problem. Because of this randomized behavior, stochastic algorithms can avoid local optima but are less reliable. Increasing

Figure 1. Processing model of cloud computing for task scheduling (Kumar et al.,2019)



the total iterations can achieve reliability. (Mirjalili et al.,2017) proposed and proved the effectiveness of SSA on small, medium, and large-scale optimization problems.

In summary, the main contribution of the proposed work is discussed as follows:

- An efficient DTC-SSA is proposed to find an efficient trade-off between makespan and execution cost.
- The proposed algorithm effectively handle deadline of user request.
- DTC-SSA decreases the deadline violation rate.
- DTC-SSA is compared with GWO and PSO in terms of makespan, total cost, and average resource utilization.

The rest of this paper is organized as follows: Section 2 represents the related work, section 3 represents the brief review of SSA, and section 4 discussed the design of the proposed algorithm. Section 5 shows the simulation setup and results of DTC-SSA and compares the existing algorithm in this section. Section 6 represents the conclusions of the paper.

2. MOTIVATION

In the commercial world, execution time and cost are two important factors that cannot be ignored, but they conflict with each other (Arabnejad et al.,2014). If someone wants to reduce the time, then more resources is needed, it attracts an extra cost, and if someone focuses on cost, it will improve execution time. Therefore, in the heterogeneous environment, the task must be scheduled so that the user's defined deadline is satisfied with the minimization of the overall cost of the task, or execution time is minimized within the user's defined budget. This paper focuses on finding a trade-off between makespan and cost while satisfying the user's deadline constraint. It also optimizes the other QoS constraint like resource utilization.

3. RELATED WORK

In cloud environment, task scheduling is a tedious issue. This problem is NP-complete and because of this, deterministic algorithms cannot provide global optimum solution effectively for large-scale problems (R.Jain et al.,2017). Here in task scheduling, many metrics are present such as execution time, resource utilization, makespan, deadline, and cost. (Iranmanesh et al.,2020) presented an improved genetic algorithm with new genetic operators to optimize cost& makespan. Scheduling results of the HEFT heuristic are taken as the initial population for DCHG-TS(deadline-constrained and cost-effective hybrid genetic algorithm). Resource utilization is improved by using a load balancing routine. (Velliangiri et al.2021) combined genetic algorithm with Electro Search and presented a Hybrid electro search with genetic algorithm (HESGA) to optimize QoS parameters like makespan and cost of the multi-cloud. Simulation results show that HESGA outperforms others.

Author (Xavier et al, 2018) proposed a chaotic social spider algorithm stimulated through the foraging nature of social spider to minimize makespan with balancing the load. They compared CSSA with other meta-heuristic, e.g., GA, ACO, PSO, ABC, and proved that CSSA is more efficient than others. Jain (Jain p. et al.,2017) presented a survey of various load balancing aware meta-heuristic algorithms. In literature (Jain R et al.) presented a systematic review of some QoS-aware nature-inspired algorithms. (Alresheedi et al.,2019) proposed a multiobjective approach that is the hybridization of salp swarm and sine-cosine algorithms (MOSSASCA) to make the most of meantime before a host shutdown (MTBHS), to decrease power utilization. It also minimizes service level agreement violations (SLAV). The performance of MOSSASCA is measure with existing multiobjective meta-heuristics, and results showed that it outperforms others.(Jain et al.,2020) proposed QoS aware task SSA to minimize makespan and improve resource utilization. Simulation result proves that it outperforms others. (ZHOU et al.,2019) proposed an empirical forecast host detection algorithm (EFA) and a weighted priority VM selection algorithm (WPA) is grounded on historical data to reduce SLA violation and energy consumption. Traces of PlanetLab as a benchmark are used as a dataset. (Awad et al.,2015) presented an enhanced PSO algorithm to achieve load balancing, reliability with minimization of makespan. Proposed algorithm identifies the failure task and schedules them on lightly loaded virtual machine. (Gill et al.,2016) proposed a PSO-based resource scheduling algorithm to scheduled heterogeneous workload. They optimized reliability, Resource utilization, and latency. (Jing et al.,2021) proposed a fault-tolerant scheduling algorithm while satisfying users' requirements for service quality. A QoS-aware discrete particle swarm optimization (QoS-DPSO) is proposed to optimize reliability under budget and time constraints.(Liu et al.,2019) presented a hardware and software collaborative optimization strategy that minimizes energy cost under deadline constraints. In this paper, an algorithm is proposed to reduce energy costs for a heterogeneous computing system. They proposed a QRLC algorithm, which combines Q learning mechanisms with RLCO (Rapid local convolution optimization) algorithm.. They also conducted real-world experiments and proved that QRLC performs better than traditional GA. (Natesan et al.,2020) presented a Performance-Cost Grey Wolf Optimization (PCGWO) algorithm to optimize resource utilization. Author scheduled tasks under deadline constraints. (Bacanin

et al.,2019) proposed GWO based scheduling algorithm. They compared proposed work with ACO, min-min, and FCFS and proved that GWO performs better than others. (Brintha et al.,2020) proposed a bat-inspired algorithm to improve resource scheduling. It minimizes execution time and improves load balancing. (Sharma et al.,2019) enhanced the ACO(EACO) algorithm by arranging incoming tasks into ascending order of length and then scheduled them into bunches. Simulation results show that EACO minimizes the makespan.

4. SALP SWARM ALGORITHM

(Mirjalili et al.,2017) proposed a swarm-inspired, Salp Swarm Algorithm (SSA). The swarming manner of salp influences SSA. Salps move in a salp chain to get effective locomotion by applying synchronized changes and foraging (Andersonet al.,1980). This salp chain is categorized into two types: Leader and Followers, where a leader is a leading salp, and followers are the remaining salps, as shown in figure 2. Leader updates its position towards the food source where a food source is continuously updated by best-obtained solution. Hence, the chain is automatically moving towards the global optimum solution, which is the objective of SSA.

Position of leader salp regarding the food source is updated as follows:

$$x_1^j = \begin{cases} FS_j + r_1\left(\left(u_j - l_j\right)r_2 + l_j\right) r_3 \geq 0.5 \\ FS_j - r_1\left(\left(u_j - l_j\right)r_2 + l_j\right) r_3 < 0.5 \end{cases} \quad (1)$$

Where p_1^j is the leader's position, FS_j is the food source, and l_j and u_j are the lower bound and upper bound, respectively in j^{th} dimension. r_2 and r_3 are random numbers generated from the interval [0,1], and r_1 is the main controlling parameter in SSA, which balances exploration and exploitation, can be calculated as follows:

$$r_1 = 2e^{-\left(\frac{4m}{maxiter}\right)^2} \quad (2)$$

Where m denotes current iteration, and maxiter is the sum of all iterations.

Position of follower salp is reorganized as follows:

$$x_i^j = \frac{1}{2}\left(x_i^j + x_{i-1}^j\right) \quad (3)$$

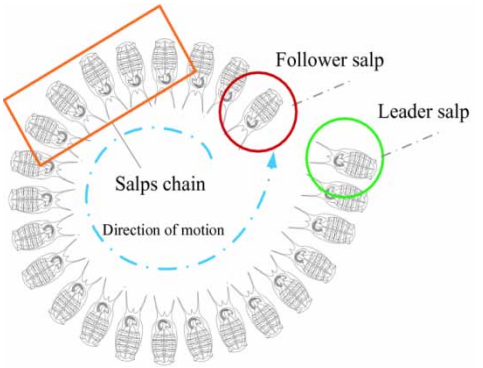
Where x_i^j is the follower's position in j^{th} dimension for $i > 1$

It can be observed from the above equations that c1 is the only controlling parameter in SSA. Similar to other swarm-inspired algorithms.

Few working criteria of SSA are as below:-

1) The algorithm stores and assigns the best result received till now to the food source, so in this manner, it preserved the best-obtained result.
2) Leader salp modifies its location with respect to the food source only, so the leader salp constantly searches the space close to it.

Figure 2. The salp chain (Faris et al.,2020)



3) Follower salps modify their location with respect to another one. In this manner, followers relocate gradually towards the leader, leading SSA from stagnating in local optima.
4) SSA has only one control parameter named C_1, so it is easy and straightforward to implement.

5. PROPOSED ALGORITHM

The mapping of cloud requests with appropriate resources is called optimized resource scheduling. These resources are provided to request based on QoS requirements mentioned by the user. It is obtained in the proposed Cost aware deadline constraint Salp Swarm Algorithm (CD-SSA).

5.1 Resource Model

This work is limited to a single data center of cloud service provider, Amazon EC2. It is assumed that the datacenter having t types of heterogeneous VM instances denoted by VMT={VMT_1, VMT_2,…….,VMT_t}. Table 1 shows the types as well as network bandwidth, RAM, CPU Cores and prices. It is assumed a global storage system with sufficient storage capacity. Processing capacity in Million Instruction Per Second(MIPS) of each VM instance is assigned as MIPS value of equivalent processor as in (Maria & Buiyya,2017). Here, we adopt VM types belong to the General Purpose instance group in US East region, which are static priced On-Demand VM provisioning model. It is also considered that at any moment there are m number of VM (VM= {Vm_1, Vm_2, ..., Vm_m}), each of them can belong to any instance type listed in VMT and they are charged as pay per use model.

5.2 User Task

Workload traces from a real system should be used to conduct experiments to make a simulation-based evaluation applicable. So, in experiments, PlanetLab data are used provided as a part of the CoMon project. In this work, CPU utilization data is obtained from more than a thousand VMs of servers at five-minute intervals, and these servers are located in more than 500 locations in the world. The data split into ten categories by the mean value of CPU utilization. Each one has single-day workload data for 500 randomly chosen servers, and each traced file have 288 readings. Here two random days are from workload traces. In the random workload, each VM runs an application with the variable workload, which is modeled to generate the utilization of CPU accordingly a uniformly distributed random variable. After generating a random function, a set of workloads automatically generate as a cloudlet called a random workload.

5.3 Problem Definition

Total Execution cost and makespan are two important QoS parameters required by cloud users. Nevertheless, both conflicts with each other, so it is a tedious task to minimize both. This work presents an efficient trade-off between total execution cost and makespan under deadline constraint.

5.3.1 Deadline Constraint

Each task t_i is associated with a firm deadline dl_i, requiring the task must be completed before the deadline.

In this work, it is assumed that all deadlines are hard deadlines and calculated by using equation 3 as follows:

$$dl_i = \beta * (minT_i + maxT_i)/2 \text{ , } \beta >= 1, i \in n \quad (4)$$

Where $MinT_i$ and $MaxT_i$ represent the minimum and maximum execution time for a given task t_i. Different value of deadline constraint is set by adjusting β (β = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10) in the experiments.

5.3.2 Makespan

It is assumed that each user's application includes a number of independent tasks having a firm deadline i.e. each task must be completed by deadline. One task requires to be accomplished in one VM instance type.

T is the set of independent tasks in the user job. Basically, they are cloud applications.

$T = \{t_1, t_2, t_3 \ldots t_n\}$ where *n* is total count of tasks.

Suppose a task t_i is assigned to virtual machine Vm_j, and $\mathcal{E}_i$ is the required execution time. The finishing time of t_i can be defined as:

$$FT(t_i) = ST(t_i) + \mathcal{E}_{ij} \quad (5)$$

Where - $ST(t_i)$ represent start time of task t_i

$\mathcal{E}_{ij}$ is the execution time of i^{th} task on j^{th} Vm can be calculated as follows:

$$\mathcal{E}_{ij} = \frac{TL_i}{capacity(Vm_j)} + \frac{TL_{i\_inputfilesize}}{Vm_j\_bw} \quad (6)$$

Where TL_i is the length of i^{th} task in MIPS, $capacity(Vm_j)$ is the total capacity of j^{th} Vm, which is calculated as

PEnum * PEmips. PEnum is expressed as the total number of process elements allocated to j^{th} Vm and PEmips is the amount of million instructions per second. $TL_{i_{inputfilesize}}$ Is the length of the input file of a task, and Vm_{j_bw} is the communication bandwidth of the j^{th} Vm.

Makespan is the execution time of last task. It can be expressed as follows:

$$Makespan = max\{FT(t_i): where\ i = 1, 2, 3, \ldots, n\} \quad (7)$$

Table 1. Instance type based on Amazon EC2[37]

VM TYPE	VCPU	Bandwidth (Mbps)	RAM(Gib)	Price ($)
a1.medium	1	10000	2	0.0255
a1.large	2	10000	4	.0510
m4.large	2	450	8	.19
m4.xlarge	4	750	16	.38
t2.small	1	600	2	.032
t2.medium	2	650	4	.0644

5.3.3 Total Cost (TC)

In this paper, it is assumed that Vm instances are charged in seconds. Thus, the total execution cost of a task on Vm is calculated by multiplying its execution time and the charge price of the VM instance.

Total cost (TC) can be calculated as:

$$\mathrm{TC} = \sum_{j=1}^{m} \mathrm{TC}_j \tag{8}$$

Here TC_j is the cost of j[th] Vm. It can be calculated as:

$$\mathrm{TC}_j = \sum_{i=1}^{n} \varepsilon_{ij} * \mathrm{P}_j \tag{9}$$

Here P_j is the price of the j[th] Vm instance and ε_{ij} is the execution time of task i that run on j[th] Vm.

To find efficient trade-off between Makespan and Total execution cost, this proposed algorithm calculates fitness function as follows:

$$\textbf{Minimize} \quad \left(TC + Makespan\right) \tag{10}$$

$$\textbf{Subject to:} \quad \mathrm{FT}(i,j) <= \mathrm{dl}_i$$

FT (i,j) is the finish time of i[th] task on j[th] Vm

In this work, the following assumptions are:

1) All tasks are independent.
2) Execution of tasks is non-preemptive.
3) All resources are non-shareable i.e., only one task can occupy one resource at a time.

5.4 Solution Initialization

This phase initializes a set of solutions. A complete solution is called a salp here. After generating a salp, its fitness is evaluated. For each task t_i in a solution, DTC-SSA finds a random Vm. If the finish time of t_i on Vm (FT (t_i,Vm)) is less than the deadline dl_i of t_i then Vm is selected otherwise, it repeats this process equal to a threshold number. This paper chooses threshold number (th) as 5. It decreases the chances of missing a deadline five times.

```
ALGORITHM 1: Solution initialization in DTC-SSA
1. For each salp S_p (p = 1 to Population Size)
2. For each t_i (i=1 to number of tasks)
3.       Find a Vm randomly from the range [1: m]
          [m: Total number of Vms]
4.          If (FT(t_i,Vm)<= dl_j)
5.             Update position of t_i
6.           Else
7.             Repeat step 3 to 7 until the threshold number. If
                 loop count is reached the threshold number, then
                 the task is marked as a failure task
8.           end if
9.         end for
10.   end for
```

5.5 Updating Positions

After each iteration, the best fitness solution is treated as leader salp, and the remaining are treated as follower salps. DTC-SSA considers a salp better if it has cost and number of missed deadline task less than other. In the Proposed algorithm, Equation 1 calculates the leader's position like SSA. After calculating, it checks the deadline constraint; if it is satisfied, then this position is updated. In SSA, the position of a follower is calculated by using equation 3, which is the position between two given positions. To improve the diversity of solution, DTC-SSA applies a random probability function shown in equation 6. Each time a random no rn is generated from the range [0-1]. If rn is less than 0.5, then the next position is x_i^j if it is greater than 0.5, then the next position is x_{i-1}^j .Otherwise, the next position is calculated by using equation 3.

$$x_i^j = \begin{cases} x_i^j & rn < 0.5 \\ x_{i-1}^j & rn > 0.5 \\ \frac{1}{2}\left(x_i^j + x_{i-1}^j\right) & rn = 0.5 \end{cases} \tag{11}$$

```
ALGORITHM 2: Position updating algorithm in CD-SSA
1.         Get salps from previous iteration
2.         for each salp  S_p
3.             if(p==1)
4.                for each task t_i
5.                    x_i^j  is calculated  by using equation 1.
6.                    If(FT(ti, x_i^j)< =dlj)
7.                        Position  is updated
8.                     Else
9.                        Position  is rejected
10.                    end if
11.                end for
12.             else
13.                for each task ti
14.                    x_i^j is calculated  by using equation 11.
```

```
15.                    If(FT(ti, x_i^j) < =dlj)
16.                        Position  is updated
17.                     Else
18.                        Position  is rejected
19.                     end if
20.                  end for
21.              end if
22.         Call algorithm 3.
23.         Calculate the fitness of each salp by using equation
10 and find the salp with minimum fitness. Replace it with the
first salp, which is treated as a leader, and remaining are
treated as followers. Assign best-obtained solution to food source
F
24.         end for
```

5.6 Rescheduling

After updating the position of each salp, this phase reschedules tasks that fail to schedule on any VM. Failure tasks are the task they missed their deadline at the time of scheduling. The objective of rescheduling is to increase the probability that the failure task can still be met its desired deadline after terminating. It minimizes the number of failure tasks with the minimum effect of makespan. For each failure task, this phase selects Vm having minimum completion time CT (Vm_j).If the deadline of failure task dl_i is greater than Vm_j then Vm_j is allocated to the task.

```
ALGORITHM 3: Rescheduling in DTC-SSA
1.         Input a Salp Sp
2.         for  each task t_i (i=1 to number of tasks)
3.            Identify the failure task t_i
4.             Determine the Vm having minimum FT (t_i,Vm)
5.            if (FT(t_i,Vm) < =dl_i)
6.               Schedule task t_i on Vm
7.            else
8.               Task t_j is marked as failure task
9.            end if
10.         end for
```

5.7 Termination

After updating the positions of leader and follower salps, the fitness value is calculated for each salp. In this proposed work, the fitness function is the sum of total cost and makespan. A salp with the best fitness is treated as a leader, and rest salps are treated as followers.

Algorithm 2 is repeated maxiter times. After termination of the process, leader salp is achieved as the best solution. Figure 3 illustrates the working of the proposed algorithm.

```
ALGORITHM 4: Proposed DTC-SSA
1.         Initialize lower boundary lb, upper boundary ub,
maximum number of iterations maxiter, Population size p, define
fitness function fitness and dimension d.
2.         Initialize Salp population S_p(p =1to population size) by
using algorithm 1 with consideration of ub and lb
3.         Repeat step 4 maxiter times.
4.         Update position of each salp based on ub and lb by
using Algorithm 2.
5.         Return F
```

6. EXPERIMENTS AND COMPARISONS

CloudSim3.0 framework is used to examine the performance of proposed algorithm. A set of experiments are conducted to assess the performance of DTC-SSA. We implemented PSO,GWO,GA and BAT algorithms, and compared their performances on CloudSim simulator.

For fair comparison, these algorithms are executed in same simulation environment. Deadlines for tasks are calculated by equation 4. In this work, hard deadlines are taken so the solutions with no deadline violations are considered only.

6.1 Experimental Setup

This study is limited to a single data center having six Vm type instances. The pricing policy, bandwidth, and RAM of a Vm are based on the Amazon EC2, as shown in table 1. Table 2 represents the parameter's initial values that are used in the simulation of algorithms. A total of 1052 user tasks are considered for this work. Population size is set as 30, and number of iterations is set as 100 for each algorithm. Each experiment is executed ten times independently. The mean values and standard deviation are obtained to evaluate the performance of algorithms. Table 3 shows the comparative results of the proposed algorithm, PSO, GWO, BAT, and GA in terms of the number of deadline violations. Experiment5.s are conducted by varying Deadline factor (β) varies from 1(tight) to 9(relaxed), whereas numbers of Vms are taken 70. Table 4 shows the comparative results of proposed DTC-SSA, PSO, GWO, BAT, and GA in terms of best, mean, and standard deviation of makespan(Q1), the Response time(Q2), Throughput(Q3), and cost(Q4), and Resource utilization(Q5). The proposed work outperforms the others in most QoS Parameters.

5.2 Experiments and Result Analysis

5.2.1 Deadline Violation

When a task missed the deadline it is called deadline violation. Total number of Vm set as 70 for this experiment. Deadline factor (β) varies from 1(tight) to 9(relaxed). Figure 3 and table 3 show that DTC-SSA minimizes the deadline violation rate as compared to others.

Makespan: It is the latest finish time of a virtual machine's task or maximum completion time. It is calculated by using equation 7. To conduct these experiments, deadline factor (β) is set as 7. Moreover, the number of Vms is varied from 100 to 500. In this, solutions with no deadline violations are considered only. It can be seen by figure 6 and table 4 that under the same deadline constraint, DTC-SSA performs better with the minimization of makespan than others.

Total Execution Cost -It is a very significant factor in task scheduling. Usually cloud user has to pay each time when he accesses cloud services. An efficient task scheduling minimizes the total cost. It is calculated by using equation 8. To conduct these experiments, the deadline factor (β) is set as 7. And the number of Vms are varied from 100 to 500. Solutions with no deadline violations are considered only. Figure 6 and 4 show that the proposed algorithm always gives minimum cost compared to PSO and GWO.

Resource utilization (RU): It can be calculated by using equation 12.

Figure 3. Flow chart of proposed DTC-SSA

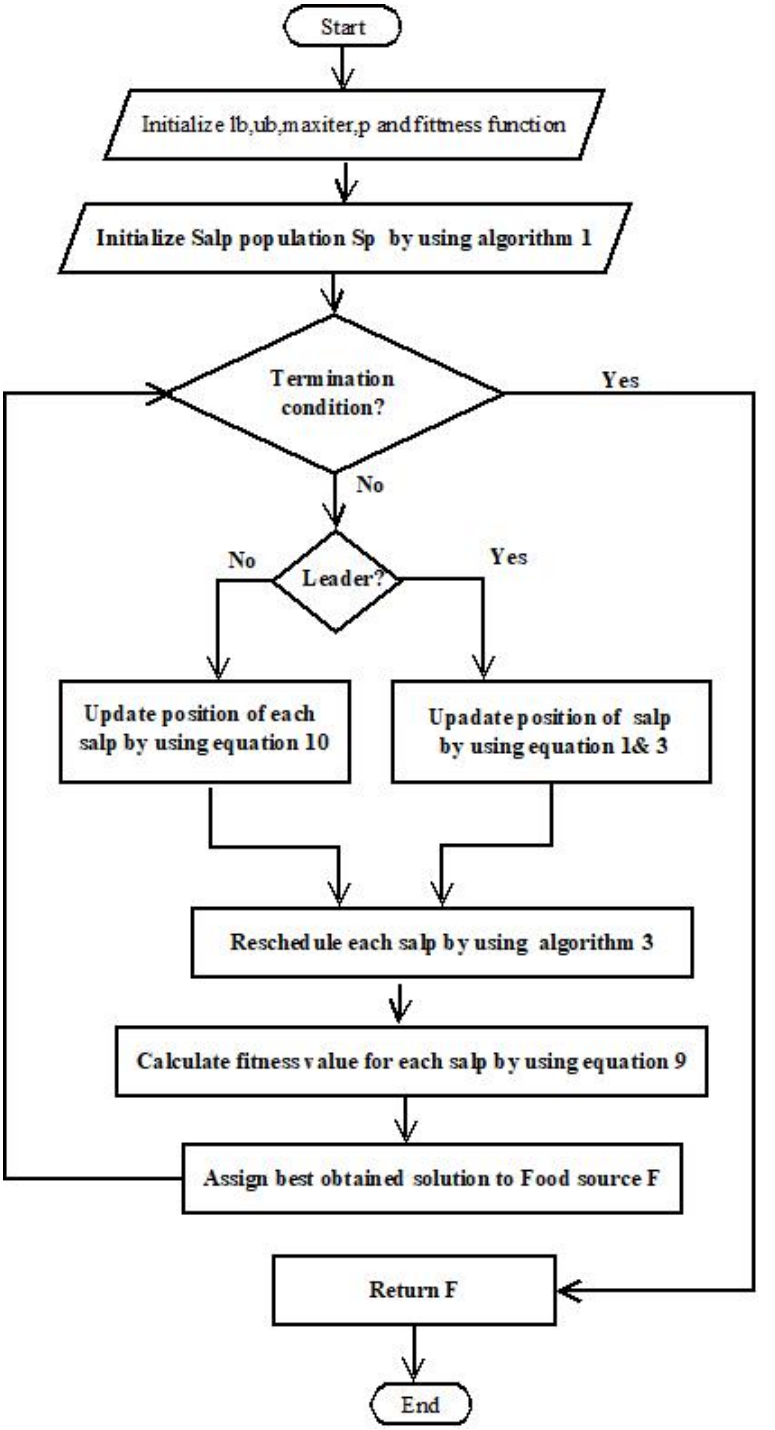


Table 2. Algorithmic parameters used in the simulation

Algorithm	Parameters	Value
DTC-SSA	random numbers r_2, r_3	generated from interval [0,1]
PSO	acceleration coefficients C_1, C_2	2
	Inertia weight factor w	.9
	random numbers r_1, r_2	generated from interval [0,1]
GWO	random numbers r_1, r_2, r_3	generated from interval [0,1]
BAT	loudness A_0	0.5026
	rate of pulse emission r_0	0.4205
	Minimum frequency Qmin	0
	Maximum frequency Qmax	2
GA	Mutation probability	generated from interval [0,1]
	random numbers r	generated from interval [0,1]
for each algorithm, population size =30, and the maximum number of iterations = 100		

Table 3. Results comparison among the algorithms in terms of number of deadline violation

Deadline factor (β)	DTC-SSA			PSO			GWO			BAT			GA		
	Best	*Mean*	*STD*	*Best*	*Mean*	*STD*	*Best*	*Mean*	*STD*	*Best*	*Mean*	*STD*	*Best*	*Mean*	*STD*
1	881	882	1.15	888	890	1.53	899	901	1.53	890	892	2.52	920	923	3
2	623	625	2.08	635	636	1	690	694	4.51	630	630	0.58	697	705	6.81
3	439	439	0.58	450	451	1.53	461	469	8.02	447	448	1.15	527	528	1.15
4	8	12	5.29	301	302	1	323	329	5.51	301	303	1.53	367	370	3.06
5	5	6	1.53	174	175	1.15	207	217	13.23	179	181	3.21	248	251	3.06
6	3	4	0.58	70	71	1.53	117	127	9.07	91	98	6.51	138	143	4.36
7	2	3	0.58	32	33	1.53	67	73	7.77	40	42	3.21	77	84	7.64
8	0	0	0	20	23	3.06	29	32	3.06	19	23	4.58	39	40	1
9	0	0	0	10	11	1.15	14	15	1.53	10	12	2.52	14	17	2.31

$$RU = \frac{\sum_{j=1}^{m} \text{Time taken by jth Vm to execute all task}}{\text{Makespan}*m} \quad (12)$$

Where m is the total number of Vm

DTC-SSA is compared with PSO and GWO based on average Resource Utilization. Figure 7 and Table 4 show that DTC-SSA outperforms others.

Throughput: Throughput is calculated using equation 13.

Table 4. Results comparison among the algorithms in terms of makespan, response time, throughput, and cost and resource utilization

No. of Vms	QoS Matrices	DTC-SSA			PSO			GWO			BAT			GA		
		Best	*Mean*	*STD*	*Best*	*Mean*	*STD*	*Best*	*Mean*	*STD*	*Best*	*Mean*	*STD*	*Best*	*Mean*	*STD*
100	Q1	7.42	7.56	0.09	7.38	7.48	0.07	9.41	9.57	0.21	8.70	9.00	0.24	8.47	8.48	0.00
	Q2	3.63	3.66	0.03	3.61	3.63	0.02	3.74	3.80	0.05	3.92	3.97	0.05	3.65	3.84	0.25
	Q3	142.29	139.28	1.86	142.45	140.54	1.46	106.69	109.80	2.20	101.29	100.52	0.54	112.99	97.46	18.06
	Q4	57.45	57.87	0.29	57.51	58.25	0.49	58.75	58.00	0.78	55.81	55.98	0.23	56.91	57.69	0.57
	Q5	0.69	0.68	0.01	0.69	0.68	0.01	0.57	0.55	0.02	0.51	0.51	0.00	0.55	0.48	0.08
200	Q1	5.93	6.73	0.57	6.88	7.23	0.25	7.00	7.25	0.20	7.00	7.34	0.25	7.08	7.34	0.19
	Q2	2.21	2.26	0.04	2.38	2.43	0.04	2.30	2.50	0.16	2.30	2.32	99.44	2.25	2.30	0.04
	Q3	177.16	156.74	13.69	152.79	145.63	5.09	150.14	145.13	4.05	141.94	130.04	7.50	148.39	136.74	10.50
	Q4	50.17	50.78	3.65	52.70	53.92	3.54	53.34	53.89	0.66	53.60	55.54	1.38	54.16	54.57	0.52
	Q5	0.45	0.40	0.03	0.39	0.38	0.01	0.37	0.33	0.05	0.37	0.34	0.02	0.38	0.35	0.02
300	Q1	5.83	6.08	0.27	6.58	7.02	0.33	6.63	6.86	0.20	5.90	6.03	0.12	6.74	6.91	0.13
	Q2	1.73	1.76	0.02	1.90	1.95	0.04	1.75	1.90	151.49	1.69	1.75	0.05	1.74	1.76	140.98
	Q3	180.50	173.16	7.53	159.65	149.92	7.17	158.59	153.33	151.43	168.36	145.18	18.87	155.89	141.95	140.19
	Q4	48.53	49.19	4.39	52.00	53.03	4.32	50.00	51.77	1.56	53.33	54.73	1.05	53.90	55.22	1.23
	Q5	0.32	0.31	0.01	0.28	0.27	0.01	0.28	0.27	0.01	0.28	0.25	0.03	0.27	0.25	0.03
400	Q1	4.98	5.41	0.29	5.62	6.01	0.24	6.06	6.20	0.16	5.29	5.38	0.09	5.49	6.09	0.53
	Q2	1.57	1.52	0.04	1.58	1.64	0.06	1.47	1.54	136.38	1.55	1.59	0.03	1.50	1.54	0.03
	Q3	183.84	194.89	10.61	186.92	175.08	7.29	173.38	169.51	4.22	153.21	142.20	11.85	175.10	158.08	12.15
	Q4	48.95	49.27	3.49	50.46	52.22	3.32	53.71	54.07	0.27	53.80	55.03	0.87	53.12	54.14	0.78
	Q5	0.24	0.26	0.02	0.24	0.23	0.01	0.22	0.22	0.00	0.20	0.18	0.02	0.22	0.20	0.01
500	Q1	4.78	5.28	0.29	5.08	5.57	0.31	5.66	5.85	0.16	4.90	5.30	0.29	5.58	5.73	0.10
	Q2	1.32	1.37	0.04	1.53	1.45	0.06	1.32	1.42	178.48	1.37	1.41	0.03	1.40	1.42	0.01
	Q3	190.64	199.67	11.80	179.01	188.34	11.18	185.72	179.83	178.41	216.17	192.28	30.90	189.76	185.21	5.43
	Q4	48.70	49.16	2.66	50.40	51.45	2.58	53.69	54.11	0.30	52.58	53.81	0.88	53.72	54.42	0.50
	Q5	0.23	0.21	0.01	0.22	0.20	0.01	0.23	0.20	0.02	0.22	0.20	0.03	0.19	0.19	0.01

Figure 4. Comparison of Deadline violation for DTC-SSA, PSO and GWO

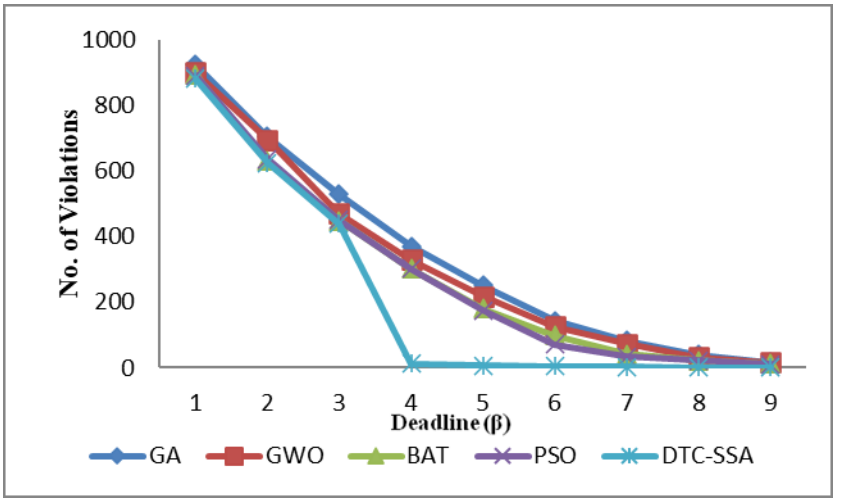
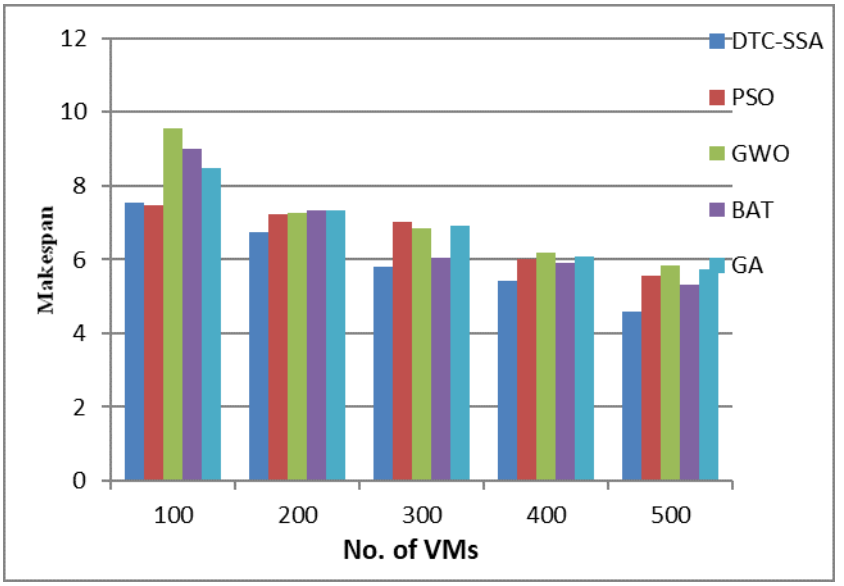


Figure 5. Comparison of Makespan for DTC-SSA, PSO, GWO, BAT and GA



$$\text{Throughput} = \frac{\text{No.of tasks completed successfully}}{\text{Makespan}} \tag{13}$$

The performance of DTC-SSA is compared with other algorithms. Figure 8 and table 4 shows that it performs better than others.

Response Time: Average Response Time is calculated using equation 14.
Average Response Time =

Figure 6. Comparison of Total Execution cost for DTC-SSA, PSO, GWO, BAT and GA

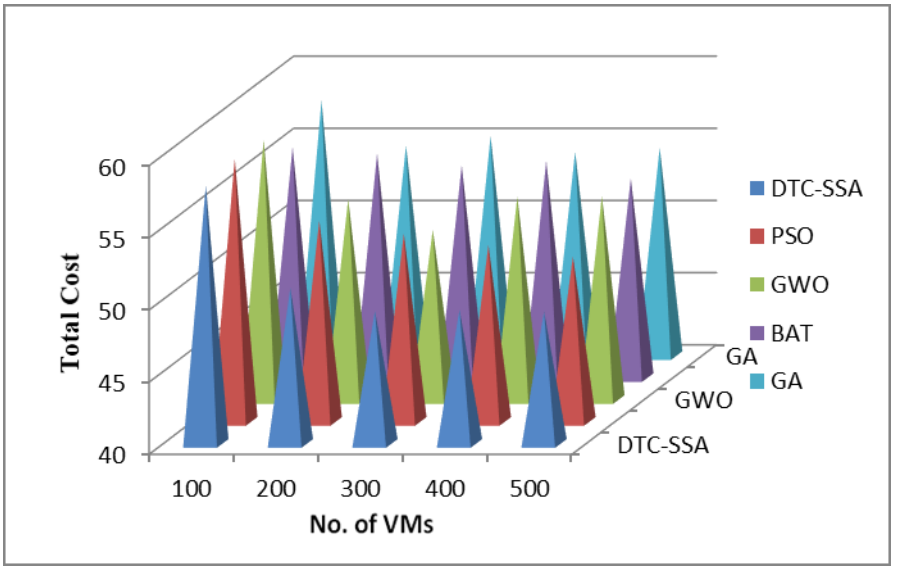


Figure 7. Comparison of Resource Utilization for DTC-SSA, PSO, GWO, BAT, and GA

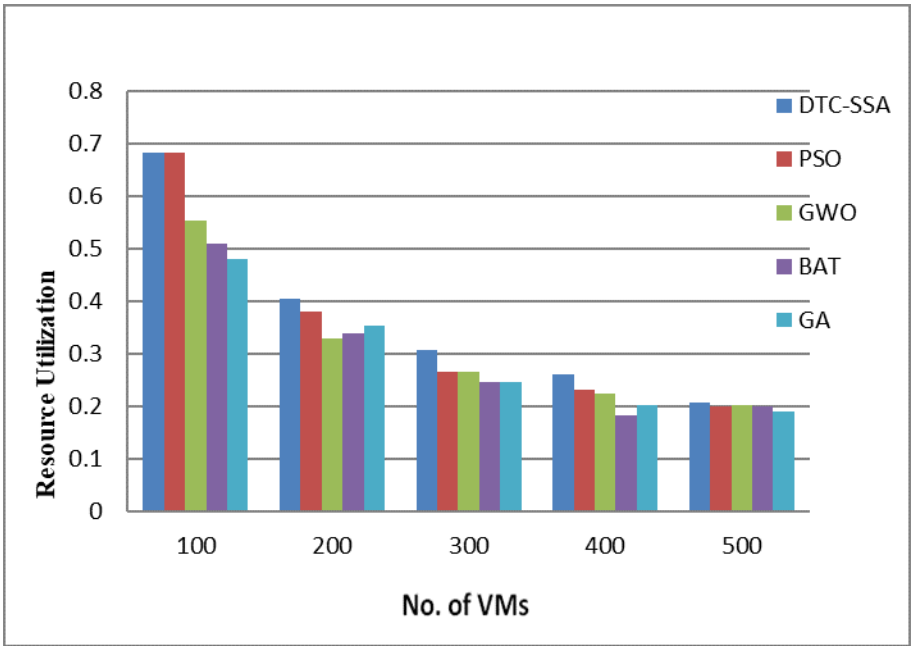
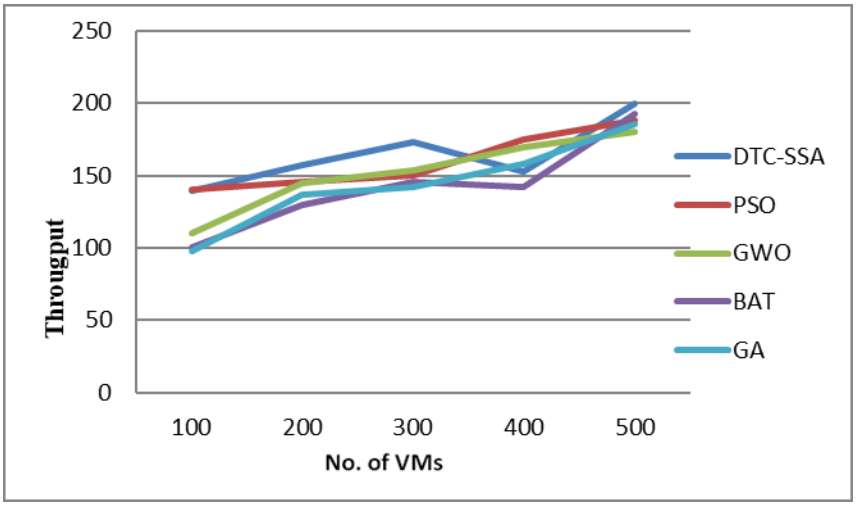


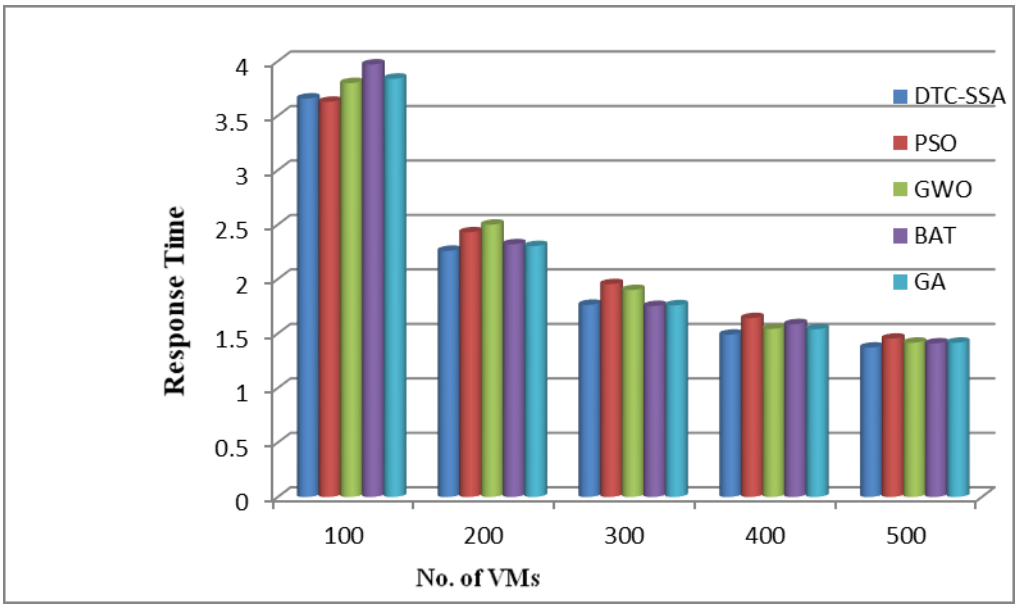
Figure 8. Comparison of Throughput for DTC-SSA, PSO, GWO, BAT and GA



$$\frac{\sum_{i=1}^{n} \text{time at which task i get started execution} - \text{arrival time of task i}}{n} \tag{14}$$

DTC-SSA is analyzed and compared with other existing algorithm for QoS parameter Response Time. It can be observed by figure 9 and table 3 that DTC-SSA outperforms others.

Figure 9. Comparison of Throughput for DTC-SSA, PSO, GWO, BAT and GA



6 CONCLUSION AND FUTURE WORK

This paper proposed a Deadline Constrained Time-Cost effective Salp Swarm Algorithm (DTC-SSA) to efficiently trade-off between cost and makespan. It is the optimal mapping of a user's request with cloud resources, also known as task scheduling. DTC-SSA minimizes the total execution cost as well as makespan while meeting the user has given deadline. It is proved by experimental results that DTC-SSA also optimizes the resource utilization as compared to PSO and GWO. It also decreases the deadline violation of incoming tasks by rescheduling. The motivation to work on these QoS constraints is that a request is completed within a given deadline with minimum cost and makespan satisfies the user more. Experimental results prove the effectiveness and efficiency of the proposed work for this problem.

In the future, we would extend the proposed algorithm by developing a hybrid that optimizes other QoS parameters like reliability availability while maintaining the SLA within user-specified deadline and budget.

REFERENCES

Alresheedi, S. S., Lu, S., Elaziz, M. A., & Ewees, A. A. (2019). Improved multiobjective salp swarm optimization for virtual machine placement in cloud computing. *Hum. Cent. Comput. Inf. Sci., 9*(15). 10.1186/s13673-019-0174-9

Anderson, P. A., & Bone, Q. (1980). Communication between individuals in salp chains II. physiology. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, *210*, 559–574.

Arabnejad, H., & Barbosa, J. G. (2014). A budget constrained scheduling algorithm for workflow applications. *Journal of Grid Computing*, *12*(4), 665–679.

Arul Xavier, V. M., & Annadurai, S. (2018). Chaotic social spider algorithm for load balance aware task scheduling in cloud computing. *Cluster Computing*. 10.1007/s10586-018-1823-x(2018)

Awad, A. I., & El-Hefnawy, N. A. (2015). Enhanced Particle Swarm Optimization For Task Scheduling In Cloud Computing Environments. *Procedia Computer Science*, *65*, 920–929.

Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I., Tuba, M., & Zivkovic, M. (2019). *Task Scheduling in Cloud Computing Environment by Grey Wolf Optimizer.* Academic Press.

Bhardwaj, S., Jain, L., & Jain, S. (2010). Cloud computing: A study of infrastructure as a service (IaaS). *Int. J. Eng. Inf. Technol.*, *2*(1), 60–63.

Bianchi, L., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, *8*(2), 239–287.

Breitgand, Maraschini, & Tordsson. (2011). *Policy-Driven Service Placement Optimization in Federated Cloud.* IBM Research Report.

Brintha, N. C., Benedict, S., & Jappes, J. T. W. (2020). Resource allocation in cloud manufacturing using bat algorithm. *Int. Journal of Manufacturing Technology Management*, *34*(3), 296–310.

Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Distributed optimization by ant colonies. *Proceedings of the first European conference on artificial life*, 134–42.

Colorni, A., Dorigo, M., Maniezzo, V., & Trubian, M. (1994). Ant System for Job-shop Scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science*, *34*(1), 39–53.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 39–43.

Geem, Z. W., Kim, J. H., & Loganathan, G. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, *76*, 60–68.

Gill, S. S., Buyya, R., Chana, I., Singh, M., & Abraham, A. (2016). BULLET: Particle Swarm Optimization Based Scheduling Technique for Provisioned Cloud Resources. *Journal of Network and Systems Management*. Advance online publication. doi:10.1007/s10922-017-9419-y

Hilliard, M. R., Liepins, G. E., & Palmer, M. (1988). Machine learning applications to job shop scheduling. *Proc. Int. Conf. Ind. Eng. Appl. Artif.Intell.Expert Systems*, 2, 728–737.

Hossam Faris, S. (2020). *Swarm Algorithm: Theory*, Literature Review, and Application in Extreme Learning Machines. Springer Nature Switzerland AG.

Iranmanesh & Naji. (n.d.). DCHG-TS: A deadline-constrained and cost-effective hybrid geneticalgorithm for scientific workflow scheduling in cloud computing. *Cluster Computing*. 10.1007/s10586-020-03145-8,2020

Jain, P., & Sharma, S. K. (2017). A systematic review of nature inspired load balancing algorithm in heterogeneous cloud computing environment. *Conference on Information and Communication Technology (CICT)*. DOI: doi:10.1109/INFOCOMTECH.2017.8340645

Jain, R. (2017). A Systematic Analysis of Nature Inspired Workflow Scheduling Algorithm in Heterogeneous Cloud Environment. *International Conference on Intelligent Communication and Computational Techniques (ICCT)*. DOI: doi:10.1109/INTELCCT.2017.8324053

Jain, R., & Sharma, N. (2021). A QoS Aware Binary Salp Swarm Algorithm for Effective Task Scheduling in Cloud Computing. In C. R. Panigrahi, B. Pati, P. Mohapatra, R. Buyya, & K. C. Li (Eds.), *Progress in Advanced Computing and Intelligent Engineering. Advances in Intelligent Systems and Computing* (Vol. 1199). Springer. https://doi.org/10.1007/978-981-15-6353-9_43

Jing, W., Zhao, C., Miao, Q., & Song, H. (2021). QoS-DPSO: QoS-aware Task Scheduling for Cloud Computing System. *Journal of Network and Systems Management*, *29*, 5. https://doi.org/10.1007/s10922-020-09573-6

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*, 459–471.

Kumar, M., & Sharma, S. C. (2019). PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing. *Neural Computing & Applications*, 1–24.

Kwok, Y. K., & Ahmad, I. (1996, May). Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, *7*(5), 506–521.

Liu, X., Liu, P., Hu, L., Zou, C., & Cheng, Z. (2019). Energy-aware task scheduling with time constraint for heterogeneous cloud datacenters. *Concurrency and Computation*.

Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., & Freund, R. F. (1999). Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, *59*(2), 107–131.

Mao, Y., Zhong, H., & Li, X. (2015). Hierarchical model-based associate tasks scheduling with the deadline constraints in the cloud. *Proceeding of the 2015 IEEE International Conference on Information and Automation.*

Mell, P., & Grance, T. (2011, June). The NIST definition of cloud computing. *Communications of the ACM*, *53*(6), 50–50.

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61.

Natesan, G., & Chokkalingam, A. (2020, January). An Improved Grey Wolf Optimization Algorithm Based Task Scheduling in Cloud Computing Environment. *The International Arab Journal of Information Technology*, *17*(1).

Panda, S. K., Gupta, I., & Jana, P. K. (2017). *Task scheduling algorithms for multi-cloud systems: allocation-aware approach*. Springer Science Business Media.

Parsa, S., & Entezari-Maleki, R. (2009). RASA: A new task scheduling algorithm in grid environment. *World Applied Sciences Journal*, *7*, 152–160.

Sakellariou, R., & Zhao, H. (2004). A hybrid heuristic for DAG scheduling on heterogeneous systems. *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 111.

Sharma, S., & Jain, R. (2019). EACO: An enhanced ant colony optimization algorithm for task scheduling in cloud computing. *International Journal of Security and Its Applications*, *13*(4), 91–100.

Toosi, A. N., Calheiros, R. N., Thulasiram, P. K., & Buyya, R. (2011). Resource provisioning policies to increase IaaS provider's profit in a federated cloud environment. *Proc. IEEE Int. Conf. High Perform.Comput. Commun.,* 279–287.

Ullman, J. D. (1975). NP-complete scheduling problems. *Journal of Computer and System Sciences*, *10*(3), 384–393.

van Laarhoven, J. M., Aarts, E. H. L., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations Research*, *40*(1), 113–125.

Velliangiri, S., & Karthikeyan, P. (2021). Hybrid electro search with genetic algorithm for task scheduling in cloud Computing. *Ain Shams Engineering Journal*, *12*, 631–639.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *EvolComput IEEE Trans*, *1*, 67–82.

Yang, X. S. (2009). Harmony search as a metaheuristic algorithm. *Stud ComputIntell, Springer, Berlin*, *191*, 1–14.

Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature inspired co- operative strategies for optimization (NICSO 2010)* (pp. 65–74). Springer.

Yang, X.-S., & Deb, S. (2009). Cuckoo search via Lévy flights. Nature & biologically inspired computing, NaBIC 2009, 210–14.

Zhang, H., Li, X., Li, H., & Huang, F. (2005). Particle swarm optimization based schemes for resource-constrained project scheduling. *Automation in Construction*, *14*(3), 393–404.

Zhou, Li, Liu, & Li. (2019). Implementation and performance analysis of various VM placement strategies in CloudSim. *IEEE Access: Practical Innovations, Open Solutions*. Advance online publication. doi:10.1109/ACCESS.2019.2899101

Richa Jain received her B.E in Computer Science and Engineering from Rajasthan University, India in 2007 and M.Tech in Computer Science and Engineering from Rajasthan Technical University, India in 2014. She is currently pursuing her Ph.D. in Computer science and Engineering in Banasthali Vidyapith, India. Her research interests include cloud computing, soft computing, algorithms, etc.

Neelam Sharma has completed her M.Tech & Ph.D in Computer Science from Banasthali Vidyapith, India. She is currently working as Associate Professor in the Department of Computer Science, Banasthali Vidyapith, India. Her teaching experience is more than 16 years. Her research interest covers Machine Learning, Pattern Recognition, Data Mining, and Image Processing. She is having 15 publications in International Conferences and Journals along with the 5 book publications.