Hybrid RSO Algorithm with SFLA for Scientific Workflow Scheduling in Cloud Using Clustering Techniques

KARPAGAM M (karpsit@gmail.com)
Amrita Vishwa Vidyapeetham Amrita School of Engineering Chennai

Research Article

Keywords: Cloud computing, workflow scheduling, NP-complete problem, metaheuristic algorithm and Shuffled Frog Leaping Algorithm (SFLA).

Posted Date: November 19th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1070996/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
Abstract

An inevitable part of the cloud computing environment is virtualization, as it can multiplex or combine many virtual machines in a single physical machine, and simultaneously an isolated environment is provided to every virtual machine. An important issue in cloud computing is workflow scheduling, which maps tasks of workflow to VMs based on various functional and non-functional requisites. Workflow scheduling is an NP-hard optimization problem and it is quite hard to achieve an optimal schedule. Metaheuristic algorithms helped in solving the problem of cloud task scheduling and this was compared to other heuristics. Reactive Search (RSO) and its structure will consist of a local heuristic based on a certain neighborhood complemented by making use of a memory-based mechanism. The Shuffled Frog Leaping Algorithm (SFLA) is based on swarm evolution that imitates information exchange divided into memeplexes when searching for food. This paper proposes a new set of optimization heuristics along with hybrid optimizations (RSO - SFLA) to solve problems in combinatorial optimization.

1. Introduction

The Internet has become the most sought-after technology which has become inevitable in day-to-day life. Many technologies that depend on the Internet such as cloud computing, World Wide Web (WWW), Internet Relay Chat (ITC) are in use for many years now. In today's computing world, cloud computing is being used commonly and can be termed as a new style of technology. This has become the technology present almost everywhere, and is a network on-demand and also user friendly (e.g, networks, servers, storage, applications, and services) that can be managed with minimal effort without much interaction with the service provider.

Cloud computing is one of the resources provided by the internet which is on an on-demand and rent basis, though the rent is quite minimal. This is one of the progressing technologies with computational and storage services as pay-as-you-go models. Generally speaking, the term internet or network is represented diagrammatically in the form of a cloud and so this technology gets its name as ‘cloud computing and it is nothing but, using the internet in providing diverse computing resources to a different sector of the population with varying needs, living at different places. The main aim of cloud computing is the distribution of work to a varying group of individuals or interaction in a conductive manner with others while using resources of other organizations which perform large-scale computation. The various disciplines of cloud computing are Load Balancing, interoperability, Virtualization, Quality of Services, and so on [2].

The software that is used to isolate the physical machine from varied virtual machines is called Virtual Machine Monitor/hypervisor. The original definition of Virtual Machine (VM) is that it is an efficient, isolated, duplicate of a real machine that allows multiplexing of the underlying physical machine. Resources are allocated to virtual machines in a fine-grained manner through virtualization. The dynamic management of workload in servers is allowed by Virtual Machine Technology with increased elasticity. A separate machine is allocated to each user application and resources are provided based on the
requirement of the application. Development platforms can be planned by the user based on the requirements of the application such as OS and other variables of the environment. Thus, multiple applications can be run in a single physical machine with different VMs. Allocation of the necessary resources is done so that there is no violation of the Service Level Agreement, even in peak demand. Full control of resource allocation is provided through virtualization to the administrator that results in the use of resources optimally [15].

One of the main activities that are being executed in the cloud computing environment is scheduling. It is done to improve work proficiency to attain the maximum benefit. The main objective of scheduling algorithms in the cloud environment is to utilize the resources properly while managing the load between resources so that the least execution time is taken.

The hybrid RSO-SFLA based method is proposed in this paper to improve workflow scheduling in the cloud [5]. The proposed method uses clustering techniques to enhance the scheduling. Related work available in the literature is presented in Section 2. The methods used were described in Section 3. The results were discussed in Section 4 and Section 5 concludes the paper.

2. Literature Review

Shirvani [12] had proposed another hybrid meta-heuristic algorithm that could solve the parallelizable and scientific workflows on an elastic cloud platform. This was because applying one single approach will not be able to get an optimal solution to these complicated issues. Several scientific workflows were modeled as a Directed Acyclic Graph (DAG) where the data dependency was in existence. For the marketplace of the cloud, every provider will deliver a set of Virtual Machine (VM) configurations that are variable. Normally, such parallelizable task scheduling based on parallel computing machines for obtaining a minimum total execution time, or a makespan will be an NP-Hard problem. For dealing with such combinatorial issues, there was a Hybrid Discrete Particle Swarm Optimization (HDPSO) algorithm that was proposed with three phases. The first phase had a random algorithm that was followed by some novel theorem applied for producing swarm members. This was an input proposed with a novel Discrete Particle Swarm Optimization (DPSO) algorithm for its second phase.

Shishido et al, [13] had examined the effects of the Particle Swarm Optimization (PSO) and the Genetic-based algorithms (GA) while attempting to optimize workflow scheduling. Another new security and cost-aware workflow algorithm were chosen for evaluating the metaheuristics and their performance. Three algorithms are evaluated in three different real-world workflows that have a constraint of risk rate ringing between 0 and 1 using a 0.1 step. The findings had shown that the GA-based algorithms were able to outperform the PSO in terms of its response time and cost-effectiveness.

Singh et al, [14] had also proposed a new workflow scheduling algorithm that was inspired by the Hybrid Chemical Reaction Optimization (HCRO) algorithm. This scheme proved to be efficient in terms of energy and also proved to have a minimized makespan. This is referred to as an Energy Efficient Workflow Scheduling (EEWS) algorithm. This EEWS was introduced in the form of a novel measure that determines
the energy that has to be conserved by taking into consideration an environment that is DVS-enabled. Employing certain simulations on various scientific applications of workflow, it had been demonstrated that the scheme proposed was able to perform better compared to the HCRO and the Multiple Priority Queues Genetic Algorithm (MPQGA). This was observed, based on different metrics of performance that include the makespan and the energy conserved. The proposed algorithm and its significance were judged using an Analysis of Variance (ANOVA) test and an LSD analysis.

Wang et al, [16] had also proposed yet another multi-objective optimization workflow scheduling approach that was based on the dynamic game-theoretic model. It aimed at reducing the workflow make spans and their total cost thereby maximizing the system fairness which was identified in terms of its workload distribution which was among certain heterogeneous cloud-based Virtual Machines (VMs). Some extensive case studies were conducted on workflow templates along with real-world third-party-based commercial IaaS clouds. The results of the experiment suggested that the approach proposed was able to outperform all the traditional ones through achieving a lower makespan of workflow, lower cost, and a better level of system fairness.

On-demand provisioning along with the availability of resources in cloud computing have made it ideal to execute the applications of the scientific workflow. The application begins execution with a minimum set of resources and some more were allocated when needed [5]. Workflow scheduling, however, was an NP-hard problem thus making solutions based on metaheuristics explored widely. Kaur and Mehta [7] had proposed yet another augmented SFLA (ASFLA) method used for resource allocation along with workflow scheduling used in IaaS. Its performance was compared to other SFLA and PSO algorithms. The efficacy of the ASFLA was assessed with some workflows using a Java-based simulator. Simulations showed a significant reduction in the cost of execution.

3. Methodology

The section presents SFLA, RSO, and the proposed hybrid RSO-for scheduling in the cloud.

3.1 Reactive Search Optimization (RSO)

Reactive Search (RS) and its structure will consist of a local heuristic based on a certain neighborhood complemented by making use of a memory-based mechanism. This had been designed to avoid cycles. There is an RSO framework that employs reinforcement learning based on history to implement knowledge acquisition by generalizing experiences before functioning. Some crucial parts are found in the RS algorithm which is: the objective function requiring optimization, the selected representation for which there are potential solutions needed. The algorithm has been designed to work for discrete problems in optimization [5].

The Shuffled Frog Leaping Algorithm (SFLA) is based on swarm evolution that imitates information exchange divided into memeplexes when searching for food. There is a need for a mixture of global search and local search in the memeplex to develop the algorithm to arrive at an optimal solution [17].
The NP-hard nature of the optimization problem in scheduling has made it very challenging to identify all the right solutions. This section can find a way in which the hybrid Genetic Algorithm (GA), Artificial Bee Colony (ABC), and the decoding heuristic can search for their ideal solutions. These decoding solutions can also generate certain feasible solutions based on tasks and their mapping to the VM [6].

There is yet another new set of optimization heuristics along with hybrid optimizations. These hybrid optimizations will assume that it has two or sometimes more than two algorithms for a similar optimization. This hybrid optimization tends to make use of a heuristic that chooses the best among such algorithms for being applied to a certain situation. For constructing this another hybrid allocator that can choose between two of the register allocation algorithms which are the linear scan and graph coloring [3] is employed. The primary goal here was the creation of another allocator to provide a proper balance between the two factors: attempting to identify a proper packing of these variables to the registers (thus achieving the good performance of running time) and reducing the allocator overheads. A hybrid optimization can bring down the effort of compilation making use of an efficient algorithm. This can be more effective but quite expensive as an optimization algorithm deems additional benefits as well.

There was one more hybrid optimization approach to design the linkages method that is applied to a dimensional synthesis and also combines the advantages of stochastic, as well as deterministic optimization. The former is based on real-valued algorithms such as the EA and is used to extensively explore the design variable space while looking for a suitable linkage [11]. The latter, on the other hand, makes use of another technique of optimization for improving efficiency through reducing its high CPU time needed by the EA techniques for such applications. For this, the deterministic approach has to be implemented in two different stages. The first one will be the evaluation of fitness in which the deterministic approach is employed to obtain a new error estimator. In the next stage, this deterministic approach further refines such solutions given by the algorithm and its evolutionary part. This new estimator ensures the evaluation of various individuals for each of these generations, thus avoiding all well-adapted linkages that cannot be detected by the other methods. This hybrid RSO-SFLA method was proposed for improving cloud workflow scheduling. The method makes use of clustering techniques for scheduling.

### 3.2 Hybrid Optimization methods

The proposed Novel Firefly Algorithm (NFA) was quite different from the Standard Firefly Algorithms (SFAs) proposed originally to solve problems of continuous optimization to map real-encoded fireflies to their solutions (task sequences). This employs a new mapping operator that is distance-based which considers the distance between the firefly and the one that is the brightest and also considers the best solution [18]. These mapped solutions are evaluated using an efficient strategy of task assignment called the Fast Task Assignment (FTA). To ensure a good start to this, an effective composite heuristic was used to generate high-quality solutions. It also develops another movement scheme to maintain a better probability in the mapping operator for inheriting the “good genes” found in the current best solution.
In the case of the standard Particle Swarm Optimization (PSO) algorithm, all initial particles have been created randomly. This randomness can also reduce the chances of converging to its best solution. The Best-Fit (BF) algorithm was merged into the PSO to improve convergence. All the other steps used in a standard PSO algorithm were retained. At the same time, the TABU Search (TS) algorithm was used as a neighborhood search algorithm to employ an intelligent search along with flexible techniques of memory. This has been designed for guiding all other algorithms to find a way to escape from the local optima [1]. Thus, this is applied for solving task scheduling problems. The TS has been thus merged to the PSO to avoid being trapped inside the local optima and accelerate the search.

The work further proposed a Multi-Objective Hybrid Artificial Bee Colony (MOHABC) algorithm that was used for Service Composition and Optimal Selection (SCOS). This was used in cloud manufacturing, where both the consumption of energy and quality of service were viewed from the perspective of the environment and economy being the two pillars in sustainable manufacturing. MOHABC makes use of the concept known as Pareto dominance for directing the search of the bee swarm. It also maintains certain non-dominated solutions in external archives. For achieving an ideal distribution of various solutions in the Pareto front, the CS along with the Levy flight was introduced.

The Ant Colony Optimization (ACO) is viewed as another improved variant of the load adjusting system of the Ant Colony and Complex Network Theory (ACCNT) while working in an open and distributed computing alliance. These calculations can make use of the pheromone of the ants to gather and further update information regarding the cloud by selecting a certain hub using a particular node for assigning this task and also evenly distributing work. For the proposed calculation, the ants begin to form their Head Hub to navigate around the system to make forward or reverse development and also locate under stacked or over-burdened hubs [9].

Firefly Algorithm (FA) is yet another very popular algorithm based on the population that can perform a global search that is suitable as it tends to have a very high degree of convergence wherein every firefly attempts to identify its best state individually. So, it can, therefore, avoid the local optimum and search for a global optimum. At the same time, Simulated Annealing (SA) algorithm tends to have a convenient process of local search. It is because of this reason that both algorithms have been combined to form an FA-SA algorithm to ideally benefit from the advantages of the algorithms and also to perform better in terms of task scheduling in the cloud. For the method proposed, the FA algorithm generally initializes first to perform a global search within the search space. Once the FA algorithm completes this, the SA algorithm will be executed to be able to perform the local search to the earlier solution that was given by the FA algorithm. This means, the initial population that was employed in the SA algorithm is not randomly chosen but obtains the value that is provided to it by the FA algorithm. This is the actual optimum value that has been given by the FA algorithm [4].

Hybrid SFLA-Genetic Algorithm has an independent local search made in every memeplex with steps that have to shuffle that is executed when the frog's interchange messages that can feature certain speedy searches similar to other algorithms. These can get trapped within the local optima and further help in
preventing and identifying an ideal global optimal solution. To ensure these traps can be avoided, there has been a need to combine the GA which can be used in the form of a substitute to update the SFLA and GA equation that is optimally combined for preempting that is being trapped within the local optima. This is effectively done at the time the GA conducts its local search. For this algorithm, each memeplex independently evolves to make a local search in different regions of its solution space after which they may be shuffled and re-divided into newer ones. This is to make global searches by exchanging information with each other [6].

There are several code models to the GA chromosome and the SFLA frogs use the codes in a model known as the real-number mode. To conduct the allocation of resources using optimal algorithms, the code model for these optimal algorithms has been made. This has further modified the position of the solution and has represented this in the form of a binary string. The length of the string was equal to the number of resources. There was a bit value of 1 that was assigned to the chosen resource indicating there was a bit value of 0 for the non-selected resource. Identified was another modification of a solution (chromosomes and frogs) that were represented in the form of a binary string for ensuring the SFLA and GA were combined to the SFLA-GA. The expressions were interchanged during selection. While conducting the evolution of the memeplexes in the SFLA, frog features represent GA chromosomes and the GA was used to conduct a local search.

**3.3 Proposed Reactive Search Optimization with Shuffled Frog Leaping Algorithm**

In this section, the RSO, along with the SFLA are discussed. In a reactive search, search history had to be employed to obtain a new parameter that was based on the feedback. This can tune all the algorithms for maintenance of its internal flexibility needed for covering various problems. This type of tuning was further automated and then implemented at the time the algorithm runs and monitors past behavior. Also, there was an automated balance that had to be made to diversification and intensification. This was for managing the dilemma known as "exploration versus exploitation" obtained by various feedback mechanisms that begin with intensification and progressively move to diversification.

The algorithms are merged to form a new hybrid SFLA–RSO algorithm. The steps involved in this are as shown below:

- Firstly, an initial population is created and each memeplex has a random position along with the cost (light intensity) is computed for the position.
- The subsequent stage is the execution of the SFLA. Here, the cost, its basis, and magnitude (light intensity) for each memeplex for all preconditions of the other memeplex are not met and are used to update the algorithm.
- Next, there may be the tendency to move towards the subsequent step that performs the RSO process. In this, the neighborhood is randomly chosen.
• The reactive hints have been a ready response to the events during alternative solutions that were tested by using an internal online feedback loop used for self-tuning in the case of crucial parameters.

• Yet another new memeplex was presented by the RSO and this was associated with the previous one. For the case of the cost, (light intensity) for a new memeplex, this was found as being better compared to the earlier one. For this, the global best was swapped to the new memeplex.

• Lastly, for a condition where the termination criterion was met, the memeplex that was the best is considered to be the output. If it is not so, all subsequent iterations begin from its memeplex algorithm from its first step.

The flowchart for the hybrid SFLA-RSO algorithm is shown in figure 1:

[IMAGE-C:\Workspace\ACDC\ImageHandler\71

4. Results And Discussion

The RSO without clustering, RSO with clustering, and SFLA-RSO Optimization methods are evaluated. Experiments are evaluated using 600 to 1000 tasks. The optimization ratio on makespan, resource utilization, and computational cost is represented in tables 5.1 to 5.4 and figures 5.2 to 5.5.

| Number of Tasks | Reactive Search Optimization without clustering | Reactive Search Optimization with clustering | SFLA-RSO Optimization |
|-----------------|-----------------------------------------------|--------------------------------------------|------------------------|
| 600             | 16                                            | 17                                         | 18                     |
| 700             | 13                                            | 17                                         | 18                     |
| 800             | 16                                            | 16                                         | 19                     |
| 900             | 16                                            | 17                                         | 19                     |
| 1000            | 16                                            | 17                                         | 20                     |

Figure 5.2 shows that the SFLA-RSO with clustering has a higher optimization ratio on makespan by 11.76% & 5.71% for 600 number of tasks, by 32.26% & 5.71% for 700 number of tasks, 17.14% & 17.14% for 800 number of tasks, 17.14% & 11.11% for 900 number of tasks and 22.22% & 16.22% for 1000 number of tasks when compared with RSO without clustering and RSO with clustering respectively.
Table 2
Resource Utilization % on SFLA-RSO with Clustering

| Number of Tasks | Reactive Search Optimization without clustering | Reactive Search Optimization with clustering | SFLA-RSO Optimization |
|-----------------|-----------------------------------------------|---------------------------------------------|------------------------|
| 600             | 85                                            | 87                                          | 88                     |
| 700             | 85                                            | 87                                          | 89                     |
| 800             | 86                                            | 82                                          | 90                     |
| 900             | 87                                            | 87                                          | 91                     |
| 1000            | 86                                            | 88                                          | 92                     |

Figure 3 shows that the SFLA-RSO with clustering has a higher Utilization ratio % by 3.47% & 1.14% for 600 number of tasks, by 4.59% & 2.27% for 700 number of tasks, 4.55% & 9.3% for 800 number of tasks, 4.49% & 4.49% for 900 number of tasks and 6.74% & 4.44% for 1000 number of tasks when compared with RSO without clustering and RSO with clustering respectively.

Table 3
Computation Cost Optimization % on SFLA-RSO with Clustering

| Number of Tasks | Reactive Search Optimization without clustering | Reactive Search Optimization with clustering | SFLA-RSO Optimization |
|-----------------|-----------------------------------------------|---------------------------------------------|------------------------|
| 600             | 4.4                                           | 4.5                                         | 4.6                    |
| 700             | 4.8                                           | 4.9                                         | 5                      |
| 800             | 4.9                                           | 5                                           | 5.1                    |
| 900             | 5.3                                           | 5.4                                         | 5.5                    |
| 1000            | 5.4                                           | 5.5                                         | 5.6                    |

Figure 4 shows that the SFLA-RSO with clustering has higher computational cost optimization % by 4.44% & 2.2% for 600 number of tasks, by 4.1% & 2.02% for 700 number of tasks, 4% & 1.98% for 800 number of tasks, 3.7% & 1.83% for 900 number of tasks and 3.64% & 1.8% for 1000 number of tasks when compared with RSO without clustering and RSO with clustering respectively.
Table 5
Makespan for SFLA-RSO with Clustering

| Number of Tasks | Reactive Search Optimization without clustering | Reactive Search Optimization with clustering | SFLA-RSO Optimization |
|-----------------|-----------------------------------------------|------------------------------------------|-----------------------|
| 600             | 63.84                                         | 63.08                                    | 62.32                 |
| 700             | 75.69                                         | 72.21                                    | 71.34                 |
| 800             | 82.32                                         | 82.32                                    | 79.38                 |
| 900             | 91.56                                         | 90.47                                    | 88.29                 |
| 1000            | 100.8                                         | 99.6                                     | 96                    |

Figure 5 shows that the SFLA-RSO with clustering has a higher optimization ratio on makespan by 2.41% & 1.21% for 600 number of tasks, by 5.92% & 1.21% for 700 number of tasks, 3.64% & 3.64% for 800 number of tasks, 3.64% & 2.44% for 900 number of tasks and 4.88% & 3.68% for 1000 number of tasks when compared with RSO without clustering and RSO with clustering respectively.

5. When The Number Of Tasks Is Increased

The RSO without clustering, RSO with clustering, and SFLA-RSO Optimization methods are evaluated. Experiments are evaluated using 5000 to 25000 tasks. The optimization ratio on makespan, resource utilization, and computational cost is represented in tables 5.5 to 5.8 and figures 5.6 to 5.9.

Table 6
Optimization Ratio on Makespan % for SFLA-RSO with Clustering

| Number of Tasks | Reactive Search Optimization without clustering | Reactive Search Optimization with clustering | SFLA-RSO Optimization |
|-----------------|-----------------------------------------------|------------------------------------------|-----------------------|
| 5000            | 16.5                                          | 17.4                                     | 18.4                  |
| 10000           | 13.3                                          | 17.4                                     | 18.5                  |
| 15000           | 16.4                                          | 16.3                                     | 19.5                  |
| 20000           | 16.3                                          | 17.3                                     | 19.6                  |
| 25000           | 16.5                                          | 17.4                                     | 20.5                  |

Figure 6 shows that the SFLA-RSO with clustering has a higher optimization ratio on makespan by 10.9% & 5.59% for 5000 number of tasks, by 32.7% & 6.13% for 10000 number of tasks, 17.3% & 17.88% for 15000 number of tasks, 18.4% & 12.47% for 20000 number of tasks and 21.6% & 16.36% for 25000 number of tasks when compared with RSO without clustering and RSO with clustering respectively.
Table 7
Resource Utilization % on SFLA-RSO with Clustering

| Number of Tasks | Reactive Search Optimization without clustering | Reactive Search Optimization with clustering | SFLA-RSO Optimization |
|-----------------|-----------------------------------------------|--------------------------------------------|------------------------|
| 5000            | 87.2                                          | 89.3                                       | 90.3                   |
| 10000           | 86.8                                          | 89.5                                       | 90.8                   |
| 15000           | 87.8                                          | 84                                         | 92.5                   |
| 20000           | 89.2                                          | 89.6                                       | 93.3                   |
| 25000           | 88.2                                          | 90                                         | 94.2                   |

Figure 7 shows that the SFLA-RSO with clustering has a higher utilization ratio % by 3.5% & 1.11% for 5000 number of tasks, by 4.5% & 1.44% for 10000 number of tasks, by 5.2% & 9.63% for 15000 number of tasks, by 4.49% & 4.05% for 20000 number of tasks and by 6.58% & 4.6% for 25000 number of tasks when compared with RSO without clustering and RSO with clustering respectively.

Table 8
Computation Cost Optimization % on SFLA-RSO with Clustering

| Number of Tasks | Reactive Search Optimization without clustering | Reactive Search Optimization with clustering | SFLA-RSO Optimization |
|-----------------|-----------------------------------------------|--------------------------------------------|------------------------|
| 5000            | 4.5                                           | 4.6                                        | 4.7                    |
| 10000           | 4.9                                           | 5                                          | 5.1                    |
| 15000           | 5                                              | 5.1                                        | 5.3                    |
| 20000           | 5.4                                           | 5.5                                        | 5.7                    |
| 25000           | 5.5                                           | 5.7                                        | 5.7                    |

Figure 8 shows that the SFLA-RSO with clustering has higher Computational cost optimization % by 4.35% & 2.2% for 5000 number of tasks, by 4% & 1.98% for 10000 number of tasks, by 5.83% & 3.85% for
15000 number of tasks, by 5.41% & 3.6% for 20000 number of tasks and by 3.6% & no change for 25000 number of tasks when compared with RSO without clustering and RSO with clustering respectively.

Table 9
Computation Cost Optimization % on SFLA-RSO with Clustering

| Number of Tasks | Reactive Search Optimization without clustering | Reactive Search Optimization with clustering | SFLA-RSO Optimization |
|-----------------|-----------------------------------------------|--------------------------------------------|-----------------------|
| 5000            | 527.72                                        | 522.032                                    | 515.712               |
| 10000           | 1107.159                                      | 1054.802                                   | 1040.755              |
| 15000           | 1608.464                                      | 1610.388                                   | 1548.82               |
| 20000           | 2199.636                                      | 2173.356                                   | 2112.912              |
| 25000           | 2683.69                                       | 2654.764                                   | 2555.13               |

Figure 9 shows that the SFLA-RSO with clustering has a higher optimization ratio on makespan by 2.3% & 1.22% for 5000 number of tasks, by 6.18% & 1.34% for 10000 number of tasks, 3.78% & 3.9% for 15000 number of tasks, 4.02% & 2.82% for 20000 number of tasks and 4.91% & 3.82% for 25000 number of tasks when compared with RSO without clustering and RSO with clustering respectively.

6. Conclusion

The Standard SFLA has a very good ability of exploration along with fast convergence speed and the traditional RSO, on the other hand, has an effective ability of exploration. At the time of applying this to solve the problems of workflow scheduling, the SFLA may be stuck to the local optima as it has a low efficiency of global exploration. The RSO had a slower speed of convergence in certain cases as there can be a lack of powerful local ability of exploitation. For overcoming these disadvantages of both algorithms, the work has proposed a new hybrid approach that was based on the RSO and the SFLA that is used for the scheduling of scientific workflows that are found in the IaaS clouds. For this, the pay-per-use pricing model. There is a decoding heuristic that is integrated into a hybrid approach that generates a feasible solution. The proposed algorithm's performance was evaluated against all other algorithms that prove its effectiveness in solving the problems of workflow scheduling in the IaaS clouds. The results proved that the SFLA-RSO along with clustering had a higher Utilization ratio % that was by about 3.47% and 1.14% for the 600 number of tasks, by about 4.59% and 2.27% for the 700 number of tasks, 4.55% and 9.3% for the 800 number of tasks, 4.49% and 4.49% for the 900 number of tasks and 6.74% and 4.44% for the 1000 number of tasks on being compared to the RSO without clustering and the RSO with clustering.

7. Future Work
In the future, other metaheuristics may be chosen from the hybrid techniques for improvement in the currently existing results.

A security mechanism can be introduced to the proposed algorithm.

**Declarations**

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

**Funding** This study was not funded by any funding agency

**Conflict of interest** The authors declare that they have no conflict of interest.

**Informed Consent** Not applicable

**Authorship Contribution** Not applicable.

**References**

1. Alkhashai, HM & Omara, FA 2016, ‘An enhanced task scheduling algorithm on cloud computing environment’, International Journal of Grid and Distributed Computing, vol. 9, no. 7, pp. 91-100.

2. Beri, R & Behal, V 2015, ‘Cloud computing: A survey on cloud Computing’, International journal of computer applications, vol. 111, no. 16, pp. 19-22.

3. Cavazos, J, Moss, JEB & O’Boyle, MF 2006, ‘Hybrid optimizations: Which optimization algorithm to use?’, in Compiler Construction: proceedings of the International Conference on Compiler Construction, Springer Berlin Heidelberg, pp. 124-138.

4. Fanian, F, Bardsiri, VK & Shokouhifar, M 2018, ‘A New Task Scheduling Algorithm using Firefly and Simulated Annealing Algorithms in Cloud Computing, (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 9, no. 2.

5. M, Geetha.K, Rajan.C (2021). A Reactive Search Optimization Algorithm for Scientific Workflow Scheduling Using Clustering Techniques. Journal of Ambient Intelligence and Humanized Computing, Springer, published, vol. 12, issue no.2.

6. M, Geetha.K, Rajan.C (2020). A modified shuffled frog leaping algorithm for scientific workflow scheduling using clustering techniques. Soft Comput 24, 637–646 (2020).

7. Kayalvili, S & Selvam, M 2018, 'Hybrid SFLA-GA algorithm for an optimal resource allocation in the cloud', Cluster Computing, vol. 22, pp. 3165–3173.

8. Kaur, D & Sharma, T 2019, ‘Scheduling Algorithms in Cloud Computing, International Journal of Computer Applications, vol. 178, no. 9, pp.16-21.

9. Navin, B & Shah, D 2017, ‘Improved Cloud Computing Using ACO Algorithm with Load Balancing’, International Journal of Innovative Research and Creative Technology IJIRCT, vol. 2, no. 6.
10. Reddy, G, Reddy, N & Phanikumar, S 2018, 'Multi-objective task scheduling using modified ant colony optimization in cloud computing, International Journal of Intelligent Engineering and Systems, vol. 11, pp. 3, pp. 242-250.

11. Shirvani, MH 2020, 'A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems', Engineering Applications of Artificial Intelligence, vol.90, no. 103501. Eusuff, M., Lansey, K, and Pasha, F.,

12. Shishido, H, Y, Estrella, J. C, Toledo, C. F. M, & Arantes, M. S 2018, 'Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds', Computers & Electrical Engineering, vol. 69, 378-394 pp. 0045-7906.

13. Singh, V, Gupta, I, & Jana, P. K 2019, 'An Energy-Efﬁcient Algorithm for Workflow Scheduling in IaaS Cloud', Journal of Grid Computing, 1-20. 1572-9184(E).

14. Sharma, V 2013, 'Virtual machine management in cloud computing environment', in Sixth International Conference on Contemporary Computing (IC3), pp. 369-374.

15. Wang, Y, Jiang, J, Xia, Y, Wu, Q, Luo, X & Zhu, Q 2018, 'A multi-stage dynamic game-theoretic approach for multi-workflow scheduling on heterogeneous virtual machines from multiple infrastructure-as-a-service clouds', in International conference on services computing, Springer, Cham, pp. 137-152.

16. Wang, Z, Zhang, D, Wang, B & Chen, W 2019, 'Research on Improved Strategy of Shuffled Frog Leaping Algorithm', in proceedings of the Thirty-Fourth Youth Academic Annual Conference of Chinese Association of Automation (YAC), IEEE, pp. 265-268.

17. Zhang, Y, Zhou, J, Sun, L, Mao, J & Sun, J 2019, 'A Novel Firefly Algorithm for Scheduling Bag-of-Tasks Applications Under Budget Constraints on Hybrid Clouds', IEEE Access, vol.7, pp. 151888-151901.

18. Zhou, J & Yao, X 2017, 'A hybrid approach combining modified artificial bee colony and cuckoo search algorithms for multi-objective cloud manufacturing service composition', International Journal of Production Research, vol.55, issue.16, pp. 4765-4784.

19. Zhou, J & Dong, S 2018, 'Hybrid glowworm swarm optimization for task scheduling in the cloud environment', Engineering Optimization, vol.50, issue.6, pp. 949-964.

**Tables**

Table 4 is not available with this version.

**Figures**
Figure 1

The flowchart for the hybrid SFLA-RSO algorithm
Figure 2

Optimization Ratio on Makespan % for SFLA-RSO with Clustering

Figure 3

Resource Utilization % on SFLA-RSO with Clustering
Figure 4

Computation Cost Optimization % on SFLA-RSO with Clustering

- Reactive Search Optimization without clustering
- Reactive Search Optimization with clustering

Figure 4

Computation Cost Optimization % on SFLA-RSO with Clustering
Figure 5
Makespan for SFLA-RSO with Clustering

Figure 6
Optimization Ratio on Makespan % for SFLA-RSO with Clustering
Figure 7
Resource Utilization % on SFLA-RSO with Clustering

Figure 8
Computation Cost Optimization % on SFLA-RSO with Clustering
Figure 9

Makespan for SFLA-RSO with Clustering