

A variant of particle swarm optimization in cloud computing environment for scheduling workflow applications

Ashish Tripathi¹, Rajnesh Singh¹, Suveg Moudgil¹, Pragati Gupta¹,
Nitin Sondhi¹, Tarun Kumar², Arun Pratap Srivastava³

¹SCSE, Galgotias University, Greater Noida, India

²Department of Computer Science and Engineering, Swami Vivekanand Subharti University, Meerut, India

³Department of Computer Science and Engineering, Lloyd Institute of Engineering and Technology, Greater Noida, India

Article Info

Article history:

Received Mar 30, 2024

Revised Nov 16, 2024

Accepted Nov 24, 2024

Keywords:

Cloud computing

PSO

Virtual machine

VPSO

Workflow scheduling

ABSTRACT

Cloud computing offers on-demand access to shared resources, with user costs based on resource usage and execution time. To attract users, cloud providers need efficient schedulers that minimize these costs. Achieving cost minimization is challenging due to the need to consider both execution and data transfer costs. Existing scheduling techniques often fail to balance these costs effectively. This study proposes a variant of the particle swarm optimization algorithm (VPSO) for scheduling workflow applications in a cloud computing environment. The approach aims to reduce both execution and communication costs. We compared VPSO with several PSO variants, including Inertia-weighted PSO, gaussian disturbed particle swarm optimization (GDPSO), dynamic-PSO, and dynamic adaptive particle swarm optimization with self-supervised learning (DAPSO-SSL). Results indicate that VPSO generally offers significant cost reductions and efficient workload distribution across resources, although there are specific scenarios where other algorithms perform better. VPSO provides a robust and cost-effective solution for cloud workflow scheduling, enhancing task-resource mapping and reducing costs compared to existing methods. Future research will explore further enhancements and additional PSO variants to optimize cloud resource management.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Arun Pratap Srivastava

Department of Computer Science and Engineering, Lloyd Institute of Engineering and Technology

Greater Noida, India

Email: apsvgi@gmail.com

1. INTRODUCTION

The term cloud computing became a buzzword in the era of network-based services. The term consists of two words: cloud and computing. The cloud represents the collection of large servers that provide services through the internet in a distributed environment [1]. These servers provide storage for data, which can be accessed and manipulated from anywhere in the world. Computing refers to processing, managing, and communicating the data stored in the cloud by using computing devices (i.e., laptops, mobile phones, and tablets). Thus, cloud computing can be understood as the on-demand availability and provisioning of various services such as storage, servers, software, hardware, and computing power over the internet with minimal, or no direct user active management. Cloud computing provides on-demand services to end-users, and it is based on a pay-as-you-go model, which means that the user has to pay only for the chosen services and not more than that. The Gmail service of Google is a real-life example of cloud computing that provides services

in terms of sending and receiving emails hosted by Google servers through any computing device and from anywhere in the world [2].

Cloud computing provides convenient, ubiquitous, and on-demand access to shared resources, with users charged based on resource usage and time spent on cloud execution. To attract users, cloud providers must offer schedulers that minimize user costs by efficiently scheduling workflows. This task is challenging due to the need to account for both execution and data transfer costs between resources. Various scheduling techniques have been employed in the past, with metaheuristic search methods being particularly popular due to their ability to converge on global optimum solutions while satisfying multiple constraints [3]-[5].

In the last few decades, the number of users on the cloud platform has increased rapidly. Due to this increase, many technical challenges have emerged for the service providers. These challenges include releasing on-demand delivery of services with a high utilization rate [6], [7]. Cloud servers execute millions of tasks simultaneously, so efficient scheduling is required. To ensure the utilization of the available resources, jobs are scheduled accordingly by the job scheduling algorithms [8], [9]. They frequently result in lower overall expenses and execution times. Workflow scheduling is a big concern in cloud computing that aims to ensure the quality-of-service requirements in terms of budget and deadline constraints while completing the execution of workflows [10]. For example, research domains such as nanoscience, cellular biology, and genome engineering use distributed datasets. These datasets are analyzed in the form of scientific workflows [11]. Workflow scheduling becomes a big issue, especially when the work needs to process large amounts of data and the computational complexity is very high. In the management of scientific workflows, cloud service providers manage the highly computationally intensive activities of scientific research. For that, the service providers use distributed resources for executing workflows as well as hide integration and orchestration details inherently [12].

In the past, several methods have been proposed for workflow scheduling on the cloud. Out of which, the application of a metaheuristic algorithm is very popular, as these approaches are very effective in providing solutions to nondeterministic polynomial-time (NP)-hard problems. Therefore, a good metaheuristic search method can produce an efficient scheduling policy that may satisfy several constraints [13], [14]. Several randomized algorithms, like genetic algorithm (GA) [15], differential evolution (DE) [16], and particle swarm optimization (PSO) [17], have been used in designing a scheduler. But due to its simplicity and popularity, PSO has been used to solve many workflow scheduling problems. Although PSO is simple to use, it has the limitation of local optimization problems. Thus, an improved solution that can overcome this limitation is required. In the proposed work, a variant of PSO named variant particle swarm optimization (VPSO) is presented. VPSO overcomes the local optima problem and provides an efficient scheduling technique [18].

VPSO will be used to design a scheduling policy that can minimize the overall cost of workflow. Execution and transfer costs are added up to create the overall cost. A task's execution cost is the time or expense associated with running the process on any server [19]. Communication/transfer cost is the cost of data transfer between two servers. VPSO minimizes the sum of these two costs with the constraint that all the resources or servers are utilized [20].

- Problem statement: effective workflow scheduling in cloud computing is essential for minimizing costs while maintaining performance. Users require low-cost solutions that account for both execution and data transfer costs. The complexity of this problem necessitates sophisticated scheduling algorithms.
- Our contribution: to address these gaps, we propose a variant of the VPSO specifically designed for cloud workflow scheduling. Our VPSO aims to minimize both execution and communication costs and efficiently manage workload distribution across resources. Unlike existing methods, our approach introduces enhancements in inertia weight adjustment and velocity calculation, leading to improved convergence on global optima.

The rest of the paper is organized into the following sections: section 2 discusses the related work. Section 3 describes the workflow scheduling problem and defines the fitness function as well. Section 4 introduces the scheduling heuristics, PSO, and VPSO. This section also discusses how the scheduling heuristic uses PSO and VPSO. Experimental work, result evaluation, and comparative analysis have been done in section 5. Finally, the paper ends with concluding words in section 6.

2. RELATED WORK

Scientific workflow over a cloud is a very challenging problem. While scheduling these workflows, we must satisfy several constraints. These constraints originate due to the large size of the tasks and the dependencies among the tasks. The recent task scheduling research activities show the presence of an NP-hard problem [21]. To solve such types of problems, heuristic approaches can be used effectively. A heuristic algorithm can be classified into two categories: 1) cluster scheduling 2) list scheduling. The clustering algorithm [22], [23] assumes that there are many processors available for executing the sub-tasks. Hence, it

uses as many processors as possible to reduce the total span of the generated scheduler. Generally, a directed cyclic graph is used to represent task dependencies in workflow applications. To schedule the workflow application, several heuristic techniques have been proposed recently. In cases where the size/quantity of the data is too large, data-intensive workflow applications are used. So, the transfer of a bulk amount of data from one computing environment to another takes much longer than usual. Also, the storage and communication costs of such data are higher than their computing costs [24].

On the other hand, list heuristic scheduling is commonly used in workflow applications. This list heuristic scheduling has two phases [25]. In phase one, some rules are applied to assign a priority to each sub-task. After that, all sub-tasks are added to the list of tasks that are already waiting for their turn in the priority queue according to their assigned priority. In phase two, the processor is assigned to the highest priority sub-task in the list that is most suitable for it. In contrast to traditional scheduling techniques, metaheuristic-based algorithms use a combinatorial process. It helps to get the optimal solution in the early stages of the generations. In each search space, such algorithms need a sufficient amount of feasible solutions to get the desired result. The algorithms, such as simulated annealing [26], ant colony optimization, particle swarm optimization [27], and genetic algorithms [28], have shown their best results in solving scheduling problems.

PSO is one of the simplest metaheuristic algorithms that has been applied to solve many real-world problems [29]. It is a population-based, nature-inspired, and global search optimization algorithm. It uses a self-adaptive technique for the survival of the swarm of particles. Like other similar algorithms, e.g., genetic algorithms, and differential evolution, PSO does not use recombination techniques directly to get the best individual in the population. PSO, on the other hand, enables the swarm's collective social behaviour to choose the best particle position [30].

3. WORKFLOW SCHEDULING PROBLEM FORMULATION

A set of tasks T has been taken $T = \{1, 2, \dots, i\}$, a set of virtual machines (VM_s) = $\{1, 2, \dots, j\}$, and a set of storage units i.e., $S = \{1, 2, \dots, k\}$. It is assumed that the average computation time of a task is T_i on a computing unit VM_j for certain known input size. Also, the unit data access cost is assumed as $D_{i,j}$. Here $D_{i,j}$ represents the cost is already known from VM_i to VM_j and $D_{i,j}=D_{j,i}$ for all $i, j \in N$, where N represents the nodes. Bandwidth between the processing units determines the communication cost. The subscription cost of the resources is decided by the service provider. Also, the communication cost of transferring data between the VMs has been charged on a per-second basis. Here, the objective is to minimize the total cost of computation by assigning the workflow task in an optimal way as shown in Figure 1. Figure 1(a) shows the workflow model with five nodes, and each node represents an individual task, which includes several instructions. While Figure 1(b) represents how VMs interact with each other for scheduling the tasks. In the workflow model, task T_1 represents the root that takes the input file, and T_5 is the last task that gives the output file. For example, T_1 produces output after completing e_{12} (T_1, e_{12}, T_2), the same concept works for all tasks such as (T_1, e_{13}, T_3) , (T_1, e_{14}, T_4) , (T_2, e_{24}, T_4) , (T_3, e_{35}, T_5) , and so on.

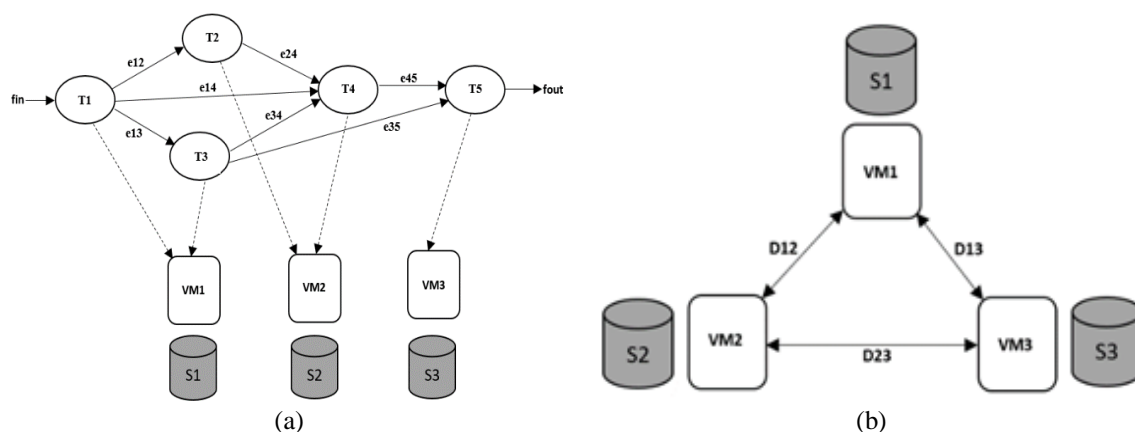


Figure 1. A workflow model example with VMs and storage (a) workflow model with five nodes and (b) workflow for scheduling tasks

4. WORKFLOW SCHEDULING BASED ON VPSO

This section presents the proposed VPSO algorithm-based workflow scheduling algorithm, which helps schedulers dynamically schedule workflow applications. The scheduling strategy is implemented with the help of the VPSO algorithm. The VPSO algorithm is created by removing the problems in the traditional PSO algorithm. The overall strategy is explained in two phases. In the first section, the proposed VPSO is explained. In section 2, scheduling methods will be discussed.

4.1. Variant particle swarm optimization

VPSO is a variant of the PSO algorithm. PSO is a population-based algorithm. It works with two vectors for solve optimization problems. The first vector is known as the position vector (X), and the other vector is known as the velocity vector (V).

In the initial generation of PSO, both position vector X and velocity vector V are initialized randomly. Since it is a population-based algorithm, a predefined number (equal to the size of the population) of position vectors and velocity vectors are initialized randomly. An n-dimensional vector $X_i=(x_{i1}, x_{i2}, \dots, x_{in})$ can be used to represent the particle position in an n-dimensional search space.

The global best position $G=(g_1, g_2, \dots, g_n)$ denotes the location of the best member of the entire swarm. According to (1) and (7), the velocity of the i^{th} particle and its new position will be determined at each step:

$$V_i = \omega * V_i + c_1 * r_1 * (P_i - X_i) + c_2 * r_2 * (G - X_i) \tag{1}$$

$$X_i = X_i + V_i \tag{2}$$

Here, the inertia weight, denoted by the symbol ω , governs how a particle's former velocity affects its present velocity. In (1), r_1 , and r_2 are the independent variables. They are distributed randomly and uniformly within a range (0, 1).

Although PSO converges very fast, it can be improved further by updating its equations. It can be seen from (2) that the position of each particle is dependent on its previous position. So if some particles of the population are close to each other, then there will be fewer changes in new positions, which will cause the particles to get stuck in the local optima and also slow down the convergence rate of the particles towards getting the optimal position. Hence, to remove this problem, we have made some changes to the PSO algorithm and created a variant of the PSO algorithm named VPSO.

In Algorithm 1, VPSO follows the model given in Figure 1 to provide the optimal mapping of all tasks to the available resources. The VPSO algorithm starts with the random initialization of the particle's position and velocity. In this workflow problem, the particles are the combination of task and resource (VM). Particles show the mapping of tasks to resources. For example, if the total number of tasks is five and computing resources are three, as shown in Figure 1. Then the particles are $T_1 \rightarrow VM_1, T_2 \rightarrow VM_2, T_3 \rightarrow VM_1, T_4 \rightarrow VM_2, T_5 \rightarrow VM_3$. This mapping is shown in Table 1. In other words, the number of jobs in the workflow determines the dimension of the particles, and the particles are the tasks that need to be assigned to the computing resources. Here, the total number of tasks is five, so the dimension of each particle is 5D.

Algorithm 1. VPSO algorithm

```

Input: set of resources  $\{VM_1, VM_2, \dots, VM_n\}$  and tasks  $T \{t_1, t_2 \dots t_n\}$ 
Output:  $g_{best}$ 
Initialize Population: number of particles equals to number of tasks(n), maxIter (maximum number of iterations)
For iter=1: maxIter
  For i = 1: n (population size)
    Compute  $C_{total}(t_i)$  using eq. (4)
    If  $C_{total}(t_i)$  for iter <  $C_{total}(t_i)$  for iter -1
      Update pbest
       $g_{best} = \min(p_{best})$ 
    Update particle position and velocity using eqn. (8) and (9)
    If stopping criteria is met
      output  $g_{best}$ 
    Else
      iter = iter+1
  
```

Table 1. A sample particle for the workflow

Tasks	T_1	T_2	T_3	T_4	T_5
Computing resources	VM_1	VM_2	VM_1	VM_2	VM_3

4.2. Scheduling strategy

The scheduling process has been implemented with the VPSO algorithm. The most important part of the cloud is the cloud scheduler, which generates a workflow schedule and satisfies certain constraints. The objective of the scheduler is to generate a workflow schedule that can minimize the overall cost of the workflow (W) application. The overall cost involves two costs: execution costs and communication costs. Since computing is taking place in a heterogeneous environment and each VM is associated with a cost. So, to calculate the overall cost of a workflow. First, the ready task will be assigned to VMs. Algorithm 2 explains the scheduling strategy.

Algorithm 2. Scheduling strategy

```

Input: W {N, E}, set of resources {VM1, VM2, ..., VMn} and tasks T {t1, t2 ... tn}
Output: gbest the workflow schedule of W over VMj
While (T)
  For i=0 to n
    For j =0 to n
      Compute Cexe and Ccomm using eq. (1) and eq. (2)
      If ∃ ((VMi ← unused) or (tparent ← VMu and tchild ← VMk))
        Compute P using eq. (3)
      Else
        P = 0
      Compute overall cost Ctotal using eq. (4)
      Assign e1i, e12 as edge weight
      Assign wij = Cexe
      WS = VPSO({tn}) ∈ T
      Use WS to assign task tj → Vj
    End
  End

```

5. EXPERIMENTAL WORK AND RESULT ANALYSIS

The research work utilized the CloudSim simulator to create a cloud environment with three VMs and five tasks. CloudSim allows for the adjustment of communication and execution costs of resources and includes a data center and a broker program for implementing scheduling algorithms. We employed the VPSO algorithm, generating particles equivalent to the number of tasks, with each particle representing a task. The dimensions of the particles corresponded to tasks, and each dimension's value indicated the VM index for task scheduling. Particles and the fitness function were implemented within a package designed for this purpose, and the broker initialized the workflow costs. A swarm object was created with the particles and fitness function, followed by the initialization of inertia and other factors. The VPSO algorithm's velocity and position equations guided the particles' evolution, resulting in a task-to-VM mapping after a specified number of iterations.

For the simulation of the cloud environment, we used the CloudSim simulator. CloudSim provides us the platform to set up an environment consisting of any number of processing units (VMs), other resources, and any number of tasks to be executed [31]. We have calculated the optimized cost and schedule for the given workflow by using the VPSO algorithm. We tested our results for a varying number of iterations, varying RAM sizes, and different costs of computing resources. The four algorithms that we used for comparison are: inertia weighted PSO, gaussian disturbed particle swarm optimization (GDPSO) [32], dynamic-PSO (DPSO) [33], dynamic adaptive particle swarm optimization with self-supervised learning (DAPSO-SSL) [34].

Table 2 and Figure 2 show how the total cost of computation varies with the range of computation costs of VMs. So, VPSO works better than the standard PSO, except for one exception. For computing the range 0.3 to 0.5, the total cost obtained by VPSO is 88,678, and by PSO is 85,625. Thus, the total cost of the VPSO is 3,053 more than the PSO. This happened because of the random traversal of particles. Also, the total cost obtained by CPSO and SOPSO is less than (by 1,406, and 2,717, respectively) VPSO.

Table 2. Cost obtained by different algorithms for different range of cost of computing resources

Cost of VM in \$	VPSO	PSO	GDPSO	DPSO	DAPSO-SSL
0.1-0.3	72,575	72,829	72,833	81,811	72,863
0.3-0.5	88,678	85,625	92,603	87,272	85,961
0.5-0.8	99,937	104,949	107,782	102,071	114,626
0.8-1.1	132,866	135,095	137,121	133,849	135,986
1.1-1.3	144,601	145,936	153,724	154,435	146,021

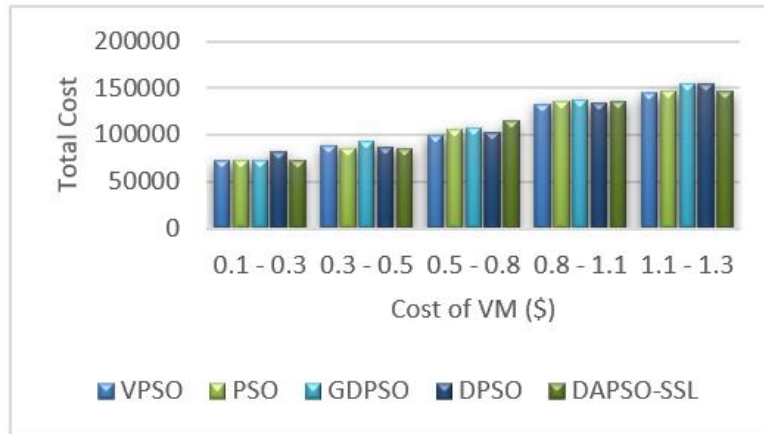


Figure 2. Cost obtained by different algorithms for different range of cost of computing resources

Table 3 and Figure 3 show how the total cost of computation varies with the number of iterations done in the algorithm. The VPSO value comes exactly below the PSO value, thus signifying an improvement in the optimization of total computation cost. Also, the cost obtained by VPSO is better than other variants of PSO. But, the performance of UWPSO and SOPSO is slightly better than that of VPSO for iterations 100 and 250, respectively. For iteration 100, the cost obtained by UWPSO is 3,265 less than VPSO, while SOPSO is 2,960 less than VPSO for 250 iterations.

Table 3. Cost obtained by different algorithms for varying number of iterations

Iterations	VPSO	PSO	GDPSO	DPSO	DAPSO-SSL
50	143,541	146,727	144,937	154,144	147,900
100	145,192	145,292	141,927	146,932	153,976
150	143,044	145,423	144,174	5,824	145,790
200	145,142	148,399	148,366	148,486	145,916
250	143,476	145,222	146,776	140,570	148,113

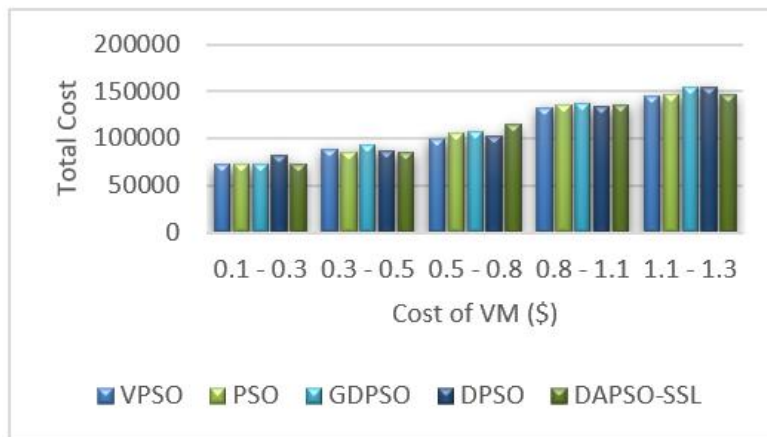


Figure 3. Cost obtained by different algorithms for varying number of iterations

Table 4 and Figure 4 show how the total cost of computation varies with the RAM size of the VMs used in the hardware. As we increase the RAM, we can see a reduction in execution time, thus reducing execution cost. The total cost is also optimized. VPSO gives better results without any exceptions. But, SOPSO shows better results than VPSO for RAM size 128. The cost obtained by SOPSO is \$568 less than that of VPSO. In this paper, a variant of PSO named VPSO is proposed to schedule the workflow problem in the cloud computing environment. Also, a new scheduling strategy has been presented to minimize the total computation cost of workflow applications.

Table 4. Cost obtained by different algorithms for varying RAM size

RAM (MB)	VPSO	PSO	GDPSO	DPSO	DAPSO-SSL
64	147493	155224	152473	150217	148359
128	147474	148584	149507	150133	147179
256	142168	143747	143180	152786	152390
512	141258	152523	148580	147251	152274
1024	142056	144113	143166	147168	142796

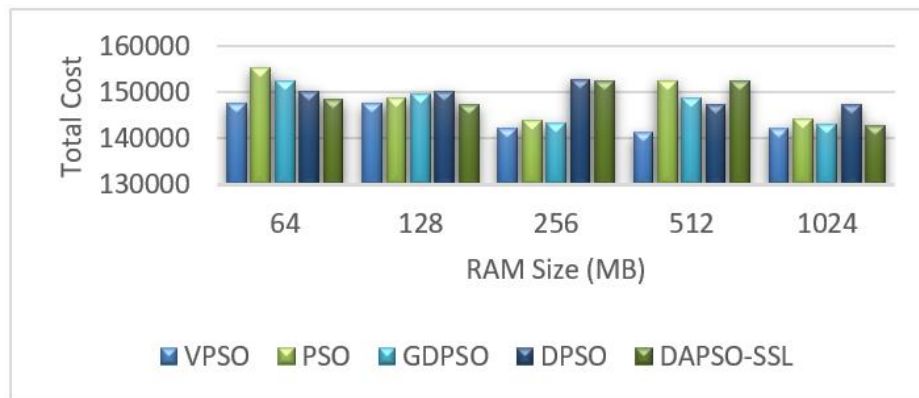


Figure 4. Cost obtained by different algorithms for varying number of iterations

We have tested the quality of the solution in every generation based on the fitness function. This fitness function includes execution costs, communication costs, and penalties to check the quality of the solution. It has been found that the proposed VPSO has provided a better task-resource mapping. The VPSO provides better cost-saving results for workflow applications as compared to the other algorithms.

6. CONCLUSION AND FUTURE WORK

Cloud computing enables on-demand access to resources, making efficient workflow scheduling critical for minimizing user costs. This study introduced a variant of the VPSO algorithm to address the challenge of balancing execution and communication costs. Our findings demonstrate that VPSO generally outperforms other PSO variants and standard scheduling algorithms in reducing costs and managing workloads effectively. Despite these few exceptions, the overall performance of VPSO in optimizing total computation cost is superior. The findings underscore the robustness of VPSO in managing workload distribution efficiently, leading to cost savings in cloud workflow scheduling. The implications of our findings are significant for cloud providers and users, offering a method to reduce operational costs while maintaining high performance.




Future work will involve testing the VPSO algorithm with other improved versions of PSO to further enhance the optimization process. Additionally, we are developing more dynamic and advanced versions of PSO to achieve even better results in scheduling workflow applications. This ongoing research aims to refine our approach and potentially extend its applicability to various cloud computing environments, ensuring scalable and efficient task-resource mapping.

REFERENCES




- [1] M. Kavis, *Architecting the cloud: design decisions for cloud computing service models (SaaS, PaaS, and IaaS)*. Wiley, 2014. doi: 10.1002/9781118691779.
- [2] M. Xin and N. Levina, "Software-as-a-service model: elaborating client-side adoption factors," *ICIS 2008 Proceedings - Twenty Ninth International Conference on Information Systems*, 2008, doi: 10.2139/ssrn.1319488.
- [3] D. T. and G. R., "Platform-as-a-service (PaaS): model and security issues," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 15, no. 1, Jul. 2015, doi: 10.11591/telkomnika.v15i1.8073.
- [4] X. Zhang, "A fine-grained task scheduling mechanism for digital economy services based on intelligent edge and cloud computing," *Journal of Cloud Computing*, vol. 12, no. 1, p. 30, Mar. 2023, doi: 10.1186/s13677-023-00402-0.
- [5] S. S. Manvi and G. K. Shyam, "Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey," *Journal of Network and Computer Applications*, vol. 41, no. 1, pp. 424–440, May 2014, doi: 10.1016/j.jnca.2013.10.004.
- [6] A. Badshah, A. Ghani, S. Shamshirband, and A. T. Chronopoulos, "Optimising infrastructure as a service provider revenue through customer satisfaction and efficient resource provisioning in cloud computing," *IET Communications*, vol. 13, no. 18, pp. 2913–2922, Nov. 2019, doi: 10.1049/iet-com.2019.0554.

- [7] P. K. Mall, A. Shukla, and J. Singh, "An approach towards early stage detection of lung cancer using machine learning," in *Lecture Notes in Networks and Systems*, vol. 676 LNNS, 2023, pp. 537–546. doi: 10.1007/978-981-99-1699-3_37.
- [8] R. Sandhu, A. Singh, M. Faiz, H. Kaur, and S. Thukral, "Enhanced text mining approach for better ranking system of customer reviews," in *Multimodal Biometric and Machine Learning Technologies: Applications for Computer Vision*, Wiley, 2023, pp. 53–69. doi: 10.1002/9781119785491.ch3.
- [9] H. K. Channi, R. Sandhu, M. Faiz, and S. M. N. Islam, "Multi-criteria decision-making approach for laptop selection: a case study," in *2023 3rd Asian Conference on Innovation in Technology, ASIANCON 2023*, IEEE, Aug. 2023, pp. 1–5. doi: 10.1109/ASIANCON58793.2023.10270203.
- [10] J. Meena, M. Kumar, and M. Vardhan, "Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint," *IEEE Access*, vol. 4, pp. 5065–5082, 2016, doi: 10.1109/ACCESS.2016.2593903.
- [11] J. T. Tsai, J. C. Fang, and J. H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm," *Computers and Operations Research*, vol. 40, no. 12, pp. 3045–3055, Dec. 2013, doi: 10.1016/j.cor.2013.06.012.
- [12] S. Chitra, B. Madhusudhanan, G. R. Sakthidharan, and P. Saravanan, "Local minima jump PSO for workflow scheduling in cloud computing environments," in *Lecture Notes in Electrical Engineering*, vol. 279 LNEE, 2014, pp. 1225–1234. doi: 10.1007/978-3-642-41674-3_170.
- [13] P. K. Mall *et al.*, "Self-attentive CNN+BERT: an approach for analysis of sentiment on movie reviews using word embedding," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 12s, pp. 612–623, 2024.
- [14] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, Jun. 2009, doi: 10.1016/j.future.2008.12.001.
- [15] H. Cheng, "A high efficient task scheduling algorithm based on heterogeneous multi-core processor," in *2010 2nd International Workshop on Database Technology and Applications, DBTA2010 - Proceedings*, IEEE, Nov. 2010, pp. 1–4. doi: 10.1109/DBTA.2010.5659041.
- [16] V. Narayan, S. Srivastava, P. K. Mall, V. Kumar, and S. Awasthi, *A theoretical analysis of simple retrieval engine*. Boca Raton: CRC Press, 2024. doi: 10.1201/9781003479031.
- [17] A. Ebaid, R. Ammar, S. Rajasekaran, and T. Fergany, "Task clustering & scheduling with duplication using recursive critical path approach (RCPA)," in *2010 IEEE International Symposium on Signal Processing and Information Technology, ISSPIT 2010*, IEEE, Dec. 2010, pp. 34–41. doi: 10.1109/ISSPIT.2010.5711720.
- [18] A. Gupta *et al.*, "ML-CPC: a pathway for machine learning based campus placement classification," *Journal of Electrical Systems*, vol. 20, no. 3s, pp. 1453–1464, Apr. 2024, doi: 10.52783/jes.1521.
- [19] G. Jaiswal, A. Uttam, D. D. Dubey, and P. K. Mall, "Resume analyser and job recommendations system based on NLP," in *2024 2nd International Conference on Disruptive Technologies, ICDT 2024*, IEEE, Mar. 2024, pp. 1584–1587. doi: 10.1109/ICDT61202.2024.10489058.
- [20] X. Liu and J. Liu, "A task scheduling based on simulated annealing algorithm in cloud computing," *International Journal of Hybrid Information Technology*, vol. 9, no. 6, pp. 403–412, Jun. 2016, doi: 10.14257/ijhit.2016.9.6.36.
- [21] S. Abdi, S. A. Motamedi, and S. Sharifian, "Task scheduling using modified PSO algorithm in cloud computing environment," in *International Conference on Machine Learning, Electrical and Mechanical Engineering (ICMLEME'2014) Jan. 8-9, 2014 Dubai (UAE)*, International Institute of Engineers, Jan. 2014, pp. 403–412. doi: 10.15242/iee.e0114078.
- [22] P. P. Adhikari, P. K. Mall, A. Mishra, and S. Srivastava, "KDSR: hybrid machine-learning solution for intrusion detection in fog computing environment," in *Lecture Notes in Networks and Systems*, vol. 1022 LNNS, 2024, pp. 393–402. doi: 10.1007/978-981-97-3601-0_28.
- [23] Z. Chenhong, Z. Shanshan, L. Qingfeng, X. Jian, and H. Jicheng, "Independent tasks scheduling based on genetic algorithm in cloud computing," in *2009 5th international conference on wireless communications, networking and mobile computing*, IEEE, Sep. 2009, pp. 1–4. doi: 10.1109/WICOM.2009.5301850.
- [24] M. Faiz, N. Fatima, and R. Sandhu, "A vaccine slot tracker model using fuzzy logic for providing quality of service," in *Multimodal Biometric and Machine Learning Technologies: Applications for Computer Vision*, Wiley, 2023, pp. 31–52. doi: 10.1002/9781119785491.ch2.
- [25] H. A. Perlin, H. S. Lopes, and T. M. Centeno, "Particle swarm optimization for object recognition in computer vision," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5027 LNAI, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 11–21. doi: 10.1007/978-3-540-69052-8_2.
- [26] S. Tiwari, K. K. Mishra, and A. K. Misra, "Test case generation for modified code using a variant of particle swarm optimization (PSO) algorithm," in *Proceedings of the 2013 10th International Conference on Information Technology: New Generations, ITNG 2013*, IEEE, Apr. 2013, pp. 363–368. doi: 10.1109/ITNG.2013.58.
- [27] J. Agrawal and S. Agrawal, "Acceleration based particle swarm optimization (APSO) for RNA Secondary structure prediction," in *Advances in Intelligent Systems and Computing*, vol. 1089, 2015, pp. 741–746. doi: 10.1007/978-3-319-08422-0_106.
- [28] K. K. Mishra, A. Tripathi, S. Tiwari, and N. Saxena, "Evolution based memetic algorithm and its application in software cost estimation," *Journal of Intelligent and Fuzzy Systems*, vol. 32, no. 3, pp. 2485–2498, Feb. 2017, doi: 10.3233/JIFS-16463.
- [29] E. Deelman *et al.*, "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, May 2015, doi: 10.1016/j.future.2014.10.008.
- [30] N. Furmento, W. Lee, A. Mayer, S. Newhouse, and J. Darlington, "ICENI: an open grid service architecture implemented with Jini," in *Proceedings of the International Conference on Supercomputing*, IEEE, 2002, pp. 37–37. doi: 10.1109/SC.2002.10027.
- [31] I. Taylor, I. Wang, M. Shields, and S. Majithia, "Distributed computing with Triana on the Grid," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 9, pp. 1197–1214, Aug. 2005, doi: 10.1002/cpe.901.
- [32] J. Cao, S. A. Jarvis, S. Saini, and G. R. Nudd, "GridFlow: workflow management for grid computing," in *Proceedings - CCGRID 2003: 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, IEEE, 2003, pp. 198–205. doi: 10.1109/CCGRID.2003.1199369.
- [33] B. Ludäscher *et al.*, "Scientific workflow management and the Kepler system," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1039–1065, Aug. 2006, doi: 10.1002/cpe.994.
- [34] T. Oinn *et al.*, "Taverna: A tool for the composition and enactment of bioinformatics workflows," *Bioinformatics*, vol. 20, no. 17, pp. 3045–3054, Nov. 2004, doi: 10.1093/bioinformatics/bth361.




BIOGRAPHIES OF AUTHORS

Dr. Ashish Tripathi    received his M.Tech degree in CSE, MNNIT Allahabad, in 2012, and a Ph.D. degree in CSE, MNNIT Allahabad, in 2015. He is currently working as an associate professor at School of Computing Science and Engineering, Galgotias University, Greater Noida, India. His research area includes optimization techniques, evolutionary algorithms, machine learning, deep learning, data science and big data analytics, cloud computing, and security. For more details, see his personal profile page at <https://sites.google.com/view/dr-ashish-tripathi/home>. He can be contacted at email: ashish.mnnit44@gmail.com.






Dr. Rajnesh Singh    received his M.Tech degree in CSE from CDAC Noida, Affiliated to Guru Gobind Singh Indraprastha University, New Delhi in 2006 and Ph.D. degree in CSE from Gautam Buddha University, Greater Noida in 2021. He is currently working as an associate professor at School of Computing Science and Engineering, Galgotias University, Greater Noida, India. He can be contacted at email: rajneshcdac.mtech@gmail.com.






Dr. Suveg Moudgil    received his M.Tech degree in CSE from Kurukshetra University, Kurukshetra, India, in 2007 and Ph.D. degree in CSE from M.M. (DU), Mullana, Ambala, Haryana, India in the area of mobile adhoc networks in 2018. He is currently working as an associate professor at the Department of Computer Science and Engineering, Galgotias University, Greater Noida, India. He can be contacted at email: suvegmodgil1@gmail.com.







Ms. Pragati Gupta    received her M.Tech degree in computer science and engineering from the Jaypee Institute of Information Technology Noida, India, in 2014, and pursuing Ph.D. in computer science and engineering from the NSUT Delhi, India. She is currently working as an assistant professor at School of Computing Science and Engineering, Galgotias University, Greater Noida, India. She can be contacted at email: cs.pragati990@gmail.com.







Mr. Nitin Sondhi    received his M.Tech from YMCA University of Science and Technology, Faridabad in 2011. He has 14+ years of experience in the teaching plus industry. He is currently working as an assistant professor at School of Computing Science and Engineering, Galgotias University, Greater Noida, India. He has published three research papers in an international journal and conference. His research area includes software engineering and machine learning. He can be contacted at email: nitin.sondhi@gmail.com.



Dr. Tarun Kumar     working as an associate professor in the Department of Computer Science and Engineering at Swami Vivekanand Subharti University Meerut (U.P.). He is in teaching profession for more than 20 years. He has presented 30 papers in national and international journals, conferences and symposiums. His area of interest includes computer network, and cryptography. He can be contacted at email: taruncdac@gmail.com.



Dr. Arun Pratap Srivastava     is a Ph.D. degree holder in computer science and engineering. He is working as a professor in the CSE Department and Dean (IQAC and research) in Lloyd Institute of Engineering and Technology. His area of interest is WSN, IoT, and taking new challenges and trying to resolve them. He can be contacted at email: apsvgi@gmail.com.