An Improved Multi-Objective Workflow Scheduling Using F-NSPSO with Fuzzy Rules

Prathibha Soma · B. Latha · V. Vijaykumar

Accepted: 6 January 2022 / Published online: 25 January 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
A lot of scientific problems in various domains from modelling sky as mosaics to understanding Genome sequencing in biological applications are modelled as workflows with a large number of interconnected tasks. Even though many works are cited in the literature on workflow scheduling, most of the existing works are focused on reducing the makespan alone. Moreover, energy efficiency is considered only in a few works included in the literature. Constraints about the dynamic workload allocation are not introduced in the existing systems. Moreover, the optimization techniques used in the existing systems have improved the QoS with little scalability in the cloud environment since they consider only the infrastructure as the service model. In this work, a new algorithm has been proposed based on the proposal of a new Multi-Objective Optimization model called F-NSPSO using NSPSO Meta-heuristics. This method allows the user to choose a suitable configuration dynamically. When compared to NSPSO an energy reduction of at least 10% has been observed for F-NSPSO for Montage, Cybershake, and Epigenomics workflow applications. Compared to the NSPSO algorithm F-NSPSO algorithm shows at least 13%, 12%, and 21% improvement in average makespan for Montage, Cybershake, and Epigenomics workflow applications respectively.

Keywords Scientific workflows · Cloud computing · Fuzzy rules · Particle swarm optimization · Energy efficiency · Makespan

Prathibha Soma
somaprathi25@gmail.com

1 Department of Information Technology, Sri Sai Ram Engineering College, Chennai, Tamil Nadu, India
2 Department of Computer Science and Engineering, Sri Sai Ram Engineering College, Chennai, Tamil Nadu, India
3 School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia
1 Introduction

Cloud computing has transformed the Information and Communication Technology (ICT) industry by delivering dynamic and highly scalable resources. It helps to reduce the high up-front investment cost for infrastructure and the maintenance and upgrade costs by allowing the organization to either outsource its computational needs or by building a private cloud data centre. Grid, distributed cluster, and cloud computing all aim to provide computational power as a utility to a large number of end users. A lot of scientific problems in various domains from modelling sky as mosaics to understanding Genome sequencing in biological applications are modelled as workflows with a large number of interconnected tasks. The scientific community is showing increasing interest in adopting the cloud platform for deploying workflow applications because of its attractive features like dynamic resource provisioning, heterogeneous resources, pay for usage of the resource, and flexible billing models. Energy consumption by IT equipment is estimated to be 40\% and 3\% respectively [1, 2] and it can also raise marginally in future. Energy consumption in data centers is mainly contributed from storage, computing, networking devices and cooling components. The other major factor contributing to the energy consumption is because of improper resource utilization. There is a high need to address this issue of energy consumption as it leads high maintenance cost for the service providers.

Scientific Workflow application contains thousands of interconnected tasks with input and output data dependence among the tasks. These applications are mathematically modelled using Directed Acyclic Graphs (DAGs), tasks are modelled as vertices of and task inter dependencies are modelled using edges. The dependencies of the DAG are stored in data structures such as 2-dimensional matrices to capture the dependencies and to store the data transfers.

2 Literature Survey

Efficient workflow scheduling with advanced optimization techniques was accomplished by employing new methods based on Direct Acyclic Graphs (DAGs), which perform scheduling operations in parallel using distributed systems. In general, for any application execution resource management and scheduling in a cloud platform is very complex and many authors have carried out their research work in this area. Figure 1 shows the Google trends graph for the research interest on the search term "Task scheduling in cloud computing."

Well established methods for optimizing workflows in distributed heterogeneous systems are available [3, 4] however they cannot be directly ported to the cloud because of the unique characteristics of cloud-like dynamic provisioning and innovative pricing models [5]. Furthermore, cloud resource scheduling is more complex than grid scheduling and involves additional variables in Service Level Agreements (SLA), policies that must be maintained between the cloud resource provider and the user. Though the cloud uses a pay-per-use model to charge the customers on their resource usage, it rounds the resource usage to the next hour. Customers will be forced to pay for resources that they did not consume. These criteria make cloud usage for scientific workload a little more complicated.

The issue of improving energy efficiency has been addressed by using techniques such as task consolidation [6], virtual machine consolidation [7, 8], and VM consolidation with a double threshold scheme. In [9–13], authors have addressed cost and
energy minimization using heuristics. Dynamic Voltage Frequency Scaling (DVFS) is applied after finding makespan using HEFT. In [14], authors have proposed multi-objective discrete particle swarm optimization with DVFS to minimize the conflicting objectives of energy consumption, makespan, and cost. In existing works, authors address minimizing energy consumption using DVFS, which may result in increase of makespan and eventually leads to SLA Violations.

Evolutionary Multi-Objective Scheduling-Cloud Platform (EMS-C) workflow scheduling algorithms [15] have been proposed for optimizing makespan and cost of workflow execution. In their work, the authors consider the cloud platform with realistic billing models. Here the authors propose modified crossover and mutation operators to address the properties of the cloud platform. They also develop a novel encoding, fitness function and population initialization procedure to address the workflow scheduling problem. The authors compare their algorithm with the other existing state-of-the-art multi-objective optimization algorithms and have proved that their model provided better performance than the existing models with respect to run time and cost. However, the authors do not consider minimization objectives from cloud providers perspective.

The problem of optimizing makespan and energy consumption for workflow applications is addressed in this work using a high performing Pareto-based Non-dominating Sorting PSO (NSPSO), which can generate all possible combinations of resource and task pair by taking into account all resource types, billing models, and granularity. The fitness function in the proposed work considers makespan, energy consumption and paid idle time of the resources. Also, because there are multiple solutions that are good in different aspects, the NSPSO fitness function includes fuzzy based decision selection in order to quickly decide on the resource, task pair that is suitable for given user preferences.

Cloud platform comprises different types of resources with different billing options. All resource and task combinations must be checked to perform efficient workflow scheduling, resulting in a large solution search space. In order to satisfy both cloud provider and scientific user requirements, a high-performing multi-objective optimization algorithm that generates all the quality solutions is required. As the set of quality solutions are huge for large-scale applications, hence an efficient fuzzy mechanism is needed which analyses these solutions quickly based on the user’s requirement.
3 Background

This section introduces the fundamental concept of multi-objective optimization. Meta heuristic techniques are increasingly being used to solve a wide range of practical problems. Traditional weighted sum approach [16] has the advantage of being simple and requiring fewer calculations to obtain the solution. However, the main disadvantage is that the weighted approach only provides a single solution, making trade-off analysis impossible. Also, it is very important to apply proper weight values based on the user preferences and a small deviation in the weight value will result in different solutions. To overcome these drawbacks, most of the multi-objective optimization problems are solved using the Pareto-optimal set which is generated by using Pareto-dominance relation. Pareto dominance relation is used for determining which optimal solution is better.

3.1 Basic Concepts of Multi-objective Optimization

Many research problems currently need optimization in more than one criterion and need the application of multi-objective optimization [17].

3.1.1 Definition of Multi-objective Optimization

Multi-objective optimization for X with vector of decision variables with m inequality constraints and q equality constraints is defined by equations given below

\[
\text{Minimize:} X(z) = [x_1(z), x_2(z), \ldots, x_n(z)]
\]

Subject to \( y_i(z) \geq 0, \quad i = 1, 2, \ldots, m \)

\( w_i(z) = 0, \quad i = 1, 2, \ldots, p \)

where \( z = [z_1, z_2, \ldots, z_k] \) and \( \text{obj} \) is the number of objective functions, \( f: \text{Robj} \rightarrow \mathbb{R} \). In these problems special operators called Pareto dominance is used for comparing the solutions in the search space.

3.1.2 Pareto Optimal Solutions

It denotes the notion of solution dominance in multi-objective optimization problems and it indicates one solution is better in objective and better or equal in another objective.

Definition 2 Pareto-Optimal Set

Formally, Pareto optimality is defined as a solution \( s \in S \) is Pareto optimal if there does not exist another solution \( s' \in S \) such that \( f(s) \prec \text{paretof}(s') \) and it is formally defined using below equation.

\[
P^* = \left\{ x \in X | \exists y \in X : f(y) \prec f(x) \right\}
\]

Springer
Definition 3 Pareto-front

Pareto front is set of all Pareto optimal points defined as

\[ PF^* = \{ f(s) = \{ f_1(s), \ldots, f_k(s) \} | x \in P^* \} \]  

3.1.3 Overview of PSO and MOPSO Meta-Heuristics

PSO [18] is a population based bio-inspired swarm-intelligent searching technique that is based on social behavior like bird flocking or fish schooling. It is an efficient metaheuristic for solving large number of NP-Hard scheduling problems in distributed systems. Detailed working of the PSO can be found in [18]. Basic PSO needs to be changed from single objective to multi-objective full-filling. The algorithm is changed to identify set of optimal solutions in each iteration and add it in the external database [17]. Algorithm 3.1 lists the steps carried out in MOPSO.

Basic PSO cannot be applied to multi-objective optimization problem because it looks for finding the single optimal solution throughout the searching process. However, in multi-objective optimization problems there is no single optimal solution and users can look for multiple solutions with trade-off among conflicting objectives. MOPSO variants available in the literature can be classified as Non-Pareto and Pareto approaches.

Algorithm 3.1 : General MOPSO

Begin

1. Create Initial Swarm

2. Initialize External Database

3. For each particle \( p_i \) in the swarm \( S \) do

   Choose the leader from the External Database

   Velocity and position are computed and updated using equations of basic PSO

End For

4. Update External Database of non-dominated solutions

5. Repeat the above steps for Max-iterations

End

3.1.4 Non-dominated Sorting PSO

The main goal of a multi-objective optimization problem is to find as many solutions that are close to the Pareto-front as possible Figure 2, shows an example of a Non-dominated Sorting technique in which three fronts, namely the first front, second front, and third front are considered.
MOPSO introduces a large number of techniques for creating selection pressure on particles to move towards the true Pareto front. One such technique called the Non-dominated sorting PSO [19] is adopted in this work. NSPSO method overcomes the drawback of the basic PSO by increasing the resource sharing among particles and their offspring. Here, N initial particles are combined with their children to create a temporary population of 2N. Amongst these 2N particles, the solutions are compared with the operators used in NSGA-II (Non-dominated Sorting Genetic Algorithm) for finding non-dominating solutions. In NSPSO, to accomplish the goal of generating large number of non-dominating solutions, swarm particles are pushed towards the true pareto optimal front by comparing all particles’ personal bests and offspring of all other particles in the entire population.

In basic PSO, at each generation t, a comparison between particle and its offspring is done which leads to losing some important non-dominating solutions.

Let \( P_t^1 \) and \( P_t^2 \) represent two particles considered in PSO algorithm at time instant t and \( F(P_t^1) \) and \( F(P_t^2) \) represent the evaluation of fitness values for particles P1 and P2. \( F(X_{t+1}^{1,1}) \) and \( F(X_{t+1}^{2,1}) \) represent fitness values for P1 and P2 at time instant t+1. In basic PSO, the comparison is performed between a particle and its offspring. For example in this scenario, the basic PSO makes the comparison between only \( F(P_t^1) \) and its offspring \( F(X_{t+1}^{1,1}) \), similarly between \( F(P_t^2) \) and its offspring \( F(X_{t+1}^{2,1}) \), which can cause losing some non-dominated solutions. This drawback is overcome in NSPSO by comparing each particle with all the offspring resulting in a comparison of 2N particles. Considering the above example, if NSPSO is applied, then a comparison is done between all the four particles and it helps in retaining useful non-dominating solutions.

Non-dominated sorting arranges the population such that the first front consists of non-dominated solutions from the population and the second front contains solutions that are dominated by the solutions of the first front only. Similarly, the solutions in the third front are dominated by the solutions of first and second and so on. A fitness value of one is assigned to the solutions of the first front, 2 for solutions of the second front, and so on. Finally, the NSPSO generates large solutions for large-scale scientific applications with different cloud resource configurations, and for scientific users choosing the desired optimal solution will be difficult. This problem has been overcome in this
proposed work by applying fuzzy rules with the triangular membership function defined in Eq. (6) for a lower limit \( l \), upper limit \( u \), and a value \( m \) which lies between \( a \) and \( b \).

\[
\mu_A(x) = \begin{cases} 
0 & x \leq l \\
\frac{x-l}{m-l} & l < x \leq m \\
\frac{u-x}{u-m} & m < x < u \\
0 & x \geq b 
\end{cases}
\]  

(6)

The basic PSO produces fast convergence which reduces the diversity of the swarm. In NSPSO, the diversity of the solutions is maintained by using the concept of niching or crowding distance calculation. In the proposed system, Niche count is used to maintain the diversity in the population.

3.1.4.1 Niche Count Calculation  
The niche count of a particle \( P_i \) indicates the closeness of the particle to its neighborhood [20]. It is calculated dynamically using Euclidean distance as given in Eq. (7)

\[
\sigma_{\text{share}} = \frac{u_2 - l_2 + u_1 - l_1}{N - 1}
\]  

(7)

In the Eq. (7), \( u_i \) and \( l_i \) represent upper and lower bounds of the objectives and \( N \) indicates swarm size. A solution with less niche count indicates less crowding and they are considered a good solution. The existing NSPSO algorithm uses the computation of Niche count for providing better solutions.

4 Proposed System

In this work, solution for optimizing scientific workflow scheduling is addressed using FNSPSO with fuzzy rules. Here, the unbounded resources with different resource types and payment methods have been considered which leads to a huge solution space for the task to resource mapping in cloud platform. This makes it difficult for scientists to choose suitable resource configurations.

In this work, the problem of optimizing makespan and energy consumption for workflow application is addressed using high-performing Pareto-based Non-dominating sorting PSO(NSPSO), which can generate all the possible combinations of resource and task pair by considering all the types, billing model and granularity of resources. In the proposed work, the fitness function takes into account makespan and energy consumption. Furthermore, because there are multiple solutions that are good in different aspects, the NSPSO fitness function includes fuzzy based decision selection in order to quickly decide which resource, task pair is suitable for given user preferences.

NSPSO algorithm has been used in [21, 22] to minimize flow time and makespan for scheduling a set of independent tasks in a distributed heterogeneous environment. In their work, the authors claim that the NSPSO generated quality solutions over W-MOPSO. In the proposed work NSPSO is used to generate Pareto-fronts for workflow scheduling problems in cloud platform. Authors in [23] have used NSPSO meta-heuristic for workflow scheduling in grid platforms. In their work, the authors used NSPSO to generate
Pareto-fronts with the goal of minimizing makespan and cost. However, the proposed work differs from the previously mentioned work in that it takes into account all of the characteristics of the cloud platform’s resources during fitness evaluation. Also, in the proposed work NSPSO is extended with fuzzy rules for selecting solutions quickly based on user preferences during fitness function evaluation.

The main objectives of the proposed system are

1. Formulating scheduling of workflow tasks [WT1, WT2, ..., WTn] to resources[R1, R2, ..., Rm] as a multi-objective optimization problem with the objective of minimizing makespan and energy consumption.
2. Solving the above problem using Pareto and Fuzzy rule based NSPSO (F-NSPSO) meta-heuristics.
3. Providing a comparative analysis of the proposed F-NSPSO algorithm with Simple DVFS, and NSPSO algorithms.

4.1 Problem Formulation

In the past, Pareto-fronts were used to represent the plot of the objective functions in scheduling problems whose non-dominated vectors are in the Pareto optimal set. In the multi-objective workflow scheduling problem with conflicting objectives, one of the requirements for the scientists is that they would be interested in getting an optimal cloud configuration with the best Virtual Machine (VM) properties, VM types, and number of VMs required for achieving an optimal makespan and energy consumption. In the proposed work, a new F-NSPSO algorithm is proposed by extending the NSPSO algorithm in which Pareto-fronts are generated for scientific workflow applications for conflicting objectives like makespan and energy consumption where fuzzy rules are used for computing the fitness. Energy consumption and makespan calculation for workflow applications in a cloud platform are explained in the following section.

4.1.1 Workflow Energy Consumption

Energy consumption E(G) for running the workflow application G in a virtualized environment is directly proportional to the number of VMs required for running the application. Total energy consumption for executing a workflow application is given by

$$E_{total}(G) = E_{computation}(G) + E_{datatransfer}(G)$$

(8)

In the above equation, $E_{computation}(G)$ represents the energy consumed for total computations of the workflow application which can be computed by finding the sum of the energy consumed by all the tasks running on all the VMs used.

In the above equation, $E_{computation}(G)$ represents the energy consumed for total computations of the workflow application which can be computed by finding the sum of the energy consumed by all the VMs used. Power consumption for executing the task on a VM depends on the power consumption model of the physical machine on which the VM is running. In the proposed work, energy consumption for the data transfer is not addressed. The power consumption of a processor in a VM consists of static and dynamic consumption [24]. As the static consumption is not of much significance, we consider only the dynamic part. Dynamic power consumption is computed using the following formula.
where $N$ represents the number of switches per clock cycle, $L$ signifies the total capacitance load, $S_{jl}$ represents the supply voltage at level $l$ on the processor $j$, and $\text{freq}$ value is the operating frequency that operates the supply voltage at level $l$. The parameters $N$ and $L$, which are device related constants, depend on each device’s capacity. The parameters $S$ and $\text{freq}$ are proportional to the computation capacity of the processor and they operate in the range of $[S_{\text{max}}, S_{\text{min}}]$ and $[\text{freq}_{\text{max}}, \text{freq}_{\text{min}}]$ respectively. Thus the energy consumption of a task $t_i$ during its runtime is calculated as follows

\[
 E_{\text{dynamic}} = P_{\text{dynamic}} \times t_{\text{exe-time}}
\]  

(10)

The above equation indicates that the energy consumed by a processor of a physical machine on which VM is running and is computed as a product of power consumption and the total time required to execute the task $t_i$. When the processor is in an idle state, it operates at $v_{\text{min}}$ and $f_{\text{min}}$ and the power consumption is computed using the following equation

\[
P_{\text{idle}} = NXLXS_{\text{freq}_{\text{min}}}^2
\]  

(11)

Energy consumption of the processor in an idle state with the time period $t_{\text{idle-time}}$ is calculated using the following formula

\[
 E_{\text{idle}} = P_{\text{idle}} \times t_{\text{idle-time}}
\]  

(12)

The total energy consumption of tasks on VM depends on the energy consumption of the processor of physical machines on which VM is running and it is defined below

\[
 E_{\text{total}}(\text{VM}) = E_{\text{dynamic}} + E_{\text{idle}}
\]  

(13)

Total power consumption by the workflow application is the sum of the power consumed by all VMs and calculated as follows

\[
 E_{\text{computations}}(G) = \sum E(\text{VM}_i)
\]  

(14)

### 4.1.2 Workflow Makespan Calculation

One more important performance measure for scientific workflow application is makespan which indicates the overall execution time of the workflow. Since workflows represent interdependent tasks, hence the makespan model is developed using a set of recurrence equations. $WF_{\text{Task start time}}(w_{t_i}, r_j)$ is calculated using Eq. 15 by choosing the maximum time between the available task and the ready task. Time for completing ready task depends on the finish time and communication cost of the task as shown in Eq. 16

\[
 WF_{\text{Task start time}}(w_{t_i}, r_j) = \left\{ \begin{array}{ll}
 0 & \text{for } w_{t_i} = w_{t_{\text{entry}}} \\
 \max\{WF_{\text{TASK avail}}(r_j), WT_{\text{ready}}(t_i, r_j)\} & \text{for } t_i \neq t_{\text{entry}}
\end{array} \right.
\]  

(15)
where $w_{tk} \in \text{Pred}(w_{ti})$, $\text{WF}_\text{TASK}	ext{finish}_{\text{time}}(w_{ti}, r_j)$ represents the end time of task $w_{ti}$ on resource $r_j$ which is calculated using Eq. (17).

$$\text{WF}_\text{TASK}	ext{finish}_{\text{time}}(w_{ti}, r_j) = \text{WF}_\text{TASK}	ext{start}_{\text{time}}(w_{ti}, r_j) + \text{WF}_\text{TASK}	ext{exe}_{\text{time}}(w_{ti}, r_j) + \text{Datatransfer}_{\text{time}}(w_{ti}, r_j)$$

Makespan of the workflow application is obtained by finding the finish time of task $w_{texit}$

$$\text{Makespan}(G) = \text{WF}_\text{TASK}	ext{finish}_{\text{time}}(w_{texit})$$

### 4.2 Pareto-Based Workflow Scheduling Using F-NSPSO

In the proposed work, non-dominated solutions generated by the NSPSO workflow scheduling algorithm are used to construct the Pareto-fronts and fuzzy rules with a newly proposed fitness function being used to perform resource and schedule optimization. Furthermore, the proposed F-NSPSO-based multi-objective optimization is useful for workflow scheduling, and the proposed model has been tested by applying the proposed algorithm to various scientific workflow applications of varying sizes. In addition, the quality of the Fuzzy and Pareto-front generated from the proposed F-NSPSO has been tested using different Pareto-front analysis metrics and the newly proposed fitness function.

#### 4.2.1 Particle Encoding and Initialization

The following steps are used to represent particles in the proposed F-NSPSO based workflow scheduling problem.

a. Apply a topological sorting algorithm to maintain task dependencies of the workflows.

b. Initialize the Virtual machine array $VM$ [Id, VM Type, Available_time]

c. Map the tasks in the sorted result to various instances for generating different particles

| Table 1 Description of virtual machine array |
|---------------------------------------------|
| ID        | Type  | Available_time |
|-----------|-------|----------------|
| VM1       | Small | 1 h            |
| VM2       | Medium| 50 min         |
| VM3       | Small | 3 h            |
| VM4       | Large | 2 h            |

Springer
Table 1 shows the sample particle encoding used in F-NSPSO for the tasks of example workflow application shown in Figure 3 and VM details given in Table 2.

4.2.2 F-NSPSO Algorithm

As the proposed system is for workflow applications where the tasks have to be executed in a predefined order, the initial population is obtained by applying a list-based HEFT algorithm. Algorithm 4.1 shows the steps used in performing workflow scheduling using F-NSPSO and the fitness function computation is given in Algorithm 4.2.
Table 2  Sample particle representation for example workflow

| VM2 (Medium, 50) | VM4 (Large, 120) | VM2 (Medium, 30) | VM1 (Small, 60) | VM3 (Small, 180) | VM3 (Small, 115) | VM4 (Large, 80) |
|------------------|------------------|------------------|-----------------|-----------------|-----------------|-----------------|
| Task_ P          | Task_ Q          | Task_ R          | Task_ S         | Task_ T         | Task_ U         | Task_ V         |
Algorithm 4.1 F-NSPSO based workflow scheduling algorithm in cloud platform

**Input:** T: Set of tasks in workflow W; R: set of VM

**Output:** Set of non-dominated solutions for workflow

S-Swarm, N-Number of particles, Max_iteration-Number of iteration,
p - particle; count=0;
PSOList→ {} , Non_dom_PSOList→ {}, p→{}

1. **Begin**
2. Initialization(W,R) //Initial particle creation
3. Fitness_function(PSOList) //Evaluate particles
4. Non_dom_PSO_List—Non_dominating_sort(PSOList)
5. **for each** particle p ∈ S **do**
6. **calculate** p.niche_count
7. **end for**(p.niche_count)
8. Non_dominating_sort(Non_dom_PSO_List)
9. **for** count < max_iterations **do**
10. **for each** particle p ∈ PSOList **do**
11. Select Gbest for p from a specified high part (5%)
12. **calculate** v_{t+1} and p_{t+1} // Calculate new velocity and position
13. New_population = P_{t+1} ∪ Pbest_t
14. **end for** (particle)
15. Non_dominating_sort(New_population) //size of the population is twice of its original
16. **for each** particle p ∈ New_population **do**
17. **calculate** p.niche_count
18. **end for**(p.niche_count)
19. Generate N solutions by choosing particles from Pareto-Fronts F1,F2,...Fn
20. Fitness_function(PSOList)
21. count=count+1
22. For each solution perform fitness analysis using fuzzy rules and find the best solution (refer table 5.3)
23. **end for** (iterations)
24. Calculate Spacing, RNI, Maximum Spread for Final populations
25. **end**
Algorithm 4.2 uses the computation of fitness values using the fitness function shown in Equation 19

$$Bt_{\text{Val}} = \frac{(W_1 \times \text{p. makespan} + W_2 \times \text{p. Power_consumed} + W_3 \times \text{p. Paid_idle_time})}{(W_1 + W_2 + W_3)}$$

where the best value is the fitness function value, computed using the weights $W_1=0.5$, $W_2=0.3$ and $W_3=0.2$. The values for the weights $W_1$, $W_2$ and $W_3$ were estimated through repeated experiments for finding the optimal values.

Table 3  Fuzzy rules to find the best solution

| Memory_size | Energy_consumption | Time to complete the workflow task | Solution_type |
|-------------|--------------------|-----------------------------------|---------------|
| Low         | High               | Large                             | Poor          |
| Low         | High               | Medium                            | Fair          |
| Low         | High               | Small                             | Very fair     |
| Low         | Average            | Large                             | Fair          |
| Low         | Average            | Medium                            | Very fair     |
| Low         | Average            | Small                             | Good          |
| Low         | Less               | Large                             | Very fair     |
| Low         | Less               | Medium                            | Good          |
| Low         | Less               | Small                             | Very good     |
| Medium      | High               | Large                             | Fair          |
| Medium      | High               | Medium                            | Very fair     |
| Medium      | Average            | Large                             | Good          |
| Medium      | Average            | Medium                            | Very fair     |
| Medium      | Less               | Large                             | Good          |
| Medium      | Less               | Small                             | Very good     |
| Medium      | Less               | Medium                            | Best          |
| High        | High               | Large                             | Very fair     |
| High        | High               | Medium                            | Good          |
| High        | Average            | Large                             | Good          |
| High        | Average            | Medium                            | Very good     |
| High        | Average            | Small                             | Best          |
| High        | Less               | Large                             | Very good     |
| High        | Less               | Medium                            | Best          |
| High        | Less               | Small                             | Excellent     |
Table 4  Parameter values used in F-NSPSO workflow scheduling

| S. No | Parameter                           | Values                                      |
|-------|-------------------------------------|---------------------------------------------|
| 1     | Number of particles[S]              | 200                                         |
| 2     | Number of iterations                | 50–200                                      |
| 3     | Inertia weight[W]                   | 0.4 to 1.0                                  |
| 4     | Random variables [C1,C2]            | 2.0                                         |
| 5     | Dimension of particles              | Number of tasks in the workflow             |

Table 5  Characteristics of real world workflow application

| Workflow  | Type           | Number of levels | Number of tasks | Number of edges | Average execution time of tasks (min) | Average data size (GB) |
|-----------|----------------|------------------|-----------------|-----------------|---------------------------------------|------------------------|
| Montage   | I/O intensive  | 7                | 25              | 95              | 8.44                                  | 3.43                   |
|           |                | 50               | 206             | 9.78            | 3.36                                  |
|           |                | 100              | 433             | 10.78           | 3.23                                  |
|           |                | 1000             | 448             | 11.36           | 3.21                                  |
| Epigenomics| Compute intensive | 8                | 24              | 75              | 681.54                                | 116.20                 |
|           |                | 46               | 148             | 844.93          | 104.81                                |
|           |                | 100              | 322             | 3954.90         | 395.10                                |
|           |                | 997              | 3228            | 3858.67         | 388.59                                |
| CyberShake| Memory intensive | 4                | 30              | 112             | 23.77                                 | 747.48                 |
|           |                | 50               | 180             | 29.32           | 864.74                                |
|           |                | 100              | 380             | 31.53           | 849.96                                |
|           |                | 100              | 3988            | 22.71           | 102.29                                |
Fig. 4  Comparison of Makespan value for Cybershake with F-NSPSO

Fig. 5  Comparison of makespan value for montage with F-NSPSO
Algorithm 4.2 Fitness Function Computing in F-NSPSO

1 Algorithm 4.2 Fitness Function(PSO_List)
2 for every ready task $t \in T$ do
3     find $t$.length, $t$.memory, $t$.input_data, $t$.output_data
4 end for
5 for each virtual machine $\in R$ do
6     find $vm$.cpu_cores, $vm$.memory, $vm$.storage,
7     $vm$.bandwidth, $vm$.mips, $vm$.available_time
8 end for
9 for each particle $p \in S$ do
10    Calculate $p$.makespan
11    Calculate $p$.energy_consumption
12    Calculate $p$.paid_idle_time
13 if $Best\_Value > High\_Value$ then
14     Fitness_Value = $Best\_Value$
15 else if $Best\_Value > Medium\_Value$ then
16     Fitness_Value = 2*$Best\_Value$
17 else // if $Best\_Value > Low\_Value$
18     Fitness_Value = 3*$Best\_Value$
19 end if
20 end for
21 return Fitness_value
22 end procedure

Fig. 6  Comparison of makespan value for epigenomics with F-NSPSO
Fig. 7 Energy consumption analysis for epigenomics workflow with F-NSPSO

Fig. 8 Energy consumption analysis for montage workflow with F-NSPSO

Fig. 9 Energy consumption analysis for Cybershake workflow with F-NSPSO
4.2.3 Fuzzy Rules

Fuzzy logic helps to perform reasoning under uncertainty. It incorporates a basic rule-based approach on well-formed formulas and the rules are represented by IF x AND y THEN z. A fuzzy inference system applies rules which are stored in a knowledge base against facts present in the application to perform inference. Table 3 shows the fuzzy rules used in the proposed algorithm for finding the best solution from a set of workflow scheduling solutions with optimal configuration from the cloud data centre in which the scheduling activities are carried out. The rules in table 3 are interpreted as IF... THEN rules, for example, the last row of the table represents the rule:

5 IF MEMORY_SIZE IS HIGH AND ENERGY_CONSUMPTION IS LESS AND AVAILABLE_TIME IS SMALL THEN SOLUTION_TYPE IS EXCELLENT

Similarly, the first row of Table 3 represents the rule:

6 IF MEMORY_SIZE IS LOW AND ENERGY_CONSUMPTION IS HIGH AND AVAILABLE_TIME IS LARGE THEN SOLUTION_TYPE IS POOR

As a result, the fuzzy inference system developed in this work employs fuzzy rules through a forward chaining inference mechanism to perform deductive inference in order to make efficient configuration and scheduling decisions. In memory size, Low memory size indicates a memory size up to 32 GB. Medium size of memory indicates a memory up to 1 TB. Finally, High memory indicates the availability of memory up to 1PB through virtual machine allocation. In the case of Energy Consumption, Less energy indicates the energy up to 1KW. Average energy consumption indicates up to 1000 KW and High energy consumption indicates up to 100 MW. Similarly for task completion time, small takes 1 hour for completion, medium takes 2 hours and high indicates 3 h. Hence, the proposed scheduling algorithm aims at optimizing these parameters in the cloud data center by applying the proposed method.

5 Results and Discussion

This section presents a description of the parameter settings used in this work to implement the proposed F-NSPSO and a set of real-world workflow applications used in the experiment. Parameters used to carry out the proposed algorithm is given in Table 4.

Table 5 lists the characteristics of various workflow applications used in the experiment.

5.1 Experiment 1: Makespan Analysis

In this experiment, the results of the FNSPSO is compared with traditional methods NSPSO and DVFS. Figures 4 and 5 show the performance of the Cybershake, and Montage applications. The experiments were repeated for varying task size of 100, 200, 300, 400 and 500. Figure 6 shows the performance of F-NSPSO for Epigenomics workflow applications with the task set 10, 20, 30, 40 and 50.

From the above figures, it can be seen that the proposed F-NSPSO performs well when it is compared with the other existing task scheduling algorithms such as DVFS, and NSPSO in terms of makespan values. Compared to the NSPSO algorithm F-NSPSO algorithm shows at least 13%, 12%, and 21% improvement in average
makespan for Montage, Cybershake, and Epigenomics workflow applications respectively. Similarly, F-NSPSO shows an improvement of at least 20% in average makespan for all three applications over DVFS algorithm.

5.2 Experiment 2: Comparison of Energy Consumption

In the second experiment, the proposed algorithm was evaluated to find its performance for energy consumption, equation 20 is used to find the improvement in energy consumption from the proposed algorithm.

$$\text{Energy Reduction} = \frac{\text{Total Energy}_{\text{DVFS or NSPSO}} - \text{Total Energy}_{\text{F-NSPSO}}}{\text{Total Energy}_{\text{DVFS or NSPSO}}}$$ (20)

An average of above 15% in the energy reduction for the proposed system over simple DVFS was achieved, the proposed system is performing better because it looks for all the combinations of available resources. Similarly, when compared to NSPSO, F-NSPSO has an energy reduction of at least 10% for all three types of workflow applications.

Figures 7, 8 and 9 show the energy consumption analysis between the proposed task scheduling algorithm F-NSPSO and the existing scheduling algorithms such as DVFS and NSPSO for Epigenomics, Montage, and Cybershake. In this case, we ran five experiments with different numbers of tasks for each application.

From the above performance analysis, it can be observed that the energy consumption of the proposed F-NSPSO is less when compared with the existing task scheduling algorithms such as DVFS and NSPSO. The improvement in the performance of the proposed algorithm can be attributed towards use of appropriate fuzzy membership function which helps in effective optimization with NSPSO.

6 Conclusion

A pareto-based solution for workflow scheduling using multi-objective optimization based on Fuzzy-NSPSO was developed in this work. The algorithm employs a fitness function to minimize the resources’ energy consumption, makespan, and paid idle time. For large-scale scientific applications, the number of non-dominated solutions will be greater; in determining the quality of the solution, memory utilization is also taken into account as another objective. In order to quickly decide the solution 16 Fuzzy rules have been developed to classify the solution and also these rules facilitate easy decision making. Performance of F-NSPSO on three real world scientific applications Epigenomics, Cybershake and Montage is done with different task sizes. F-NSPSO has given better performance compared to DVFS and NSPSO.

Funding None.

Data Availability Yes

Code Availability Yes
Declarations

Conflict of interest All authors declares that they have no conflict of interest.

Animal and Human Rights Additional declarations for articles in life science journals that report the results of studies involving humans and/or animals

References

1. Delforge, P., & Whitney, J. (2014). Issue paper: Data centre efficiency assessment scaling up energy efficiency across the data centre industry: Evaluating key drivers and barriers. Natural Resources Defense Council (NRDC).
2. Sareh, F. P. (2016). *Energy-efficient management of resources in enterprise and container based clouds*. The University of Melbourne.
3. Dong, F., & Selimi, G.A. (2006). Scheduling algorithms for grid computing: State of the art and open problems. School of Computing, Queen’s University, Kingston, Ontario. Technical Report No. 2006-504.
4. Yu, J., Buyya, R., & Ramamohanarao, K. (2008). ‘Workflow scheduling algorithms for grid computing’, Metaheuristics for scheduling in distributed computing environments, studies. *Computational Intelligence, 146*, 173–214.
5. Abrishami, S., Naghibzadeh, M., & Epema, D. H. (2013). Deadline constrained workflow scheduling algorithms for infrastructure as a service clouds. *Future Generation Computer Systems, 29*(1), 158–169.
6. Ferdaus, M. H., Murshed, M. M., Calheiros, R. N., & Buyya, R. (2014). Virtual machine consolidation in cloud data centers using aco metaheuristic. In *Proceedings of European conference on parallel processing*, Springer, pp. 306–317.
7. Lee, Y. C., & Zomaya, A. Y. (2012). Energy efficient utilization of resources in cloud computing systems. *Journal of the Supercomputing, 60*(2), 268–280.
8. Mohanapiya, N., Kousalya, G., Balakrishnan, P., & Pethuru Raj, C. (2018). Energy efficient workflow scheduling with virtual machine consolidation for green cloud computing. *Journal of Intelligent & Fuzzy Systems, 34*(3), 1561–1572.
9. Li, J., Li, Y. K., Chen, X., Lee, P. P., & Lou, W. (2015). A hybrid cloud approach for secure authorized deduplication. *IEEE Transactions on Parallel and Distributed Systems, 26*(5), 1206–1216.
10. Pietri, I., & Sakellariou, R. (2014) Cost-efficient provisioning of cloud resources priced by CPU frequency. In *Proceedings of the IEEE/ACM Seventh international conference on utility and cloud computing*, pp. 483–484.
11. Tang, Z., Qi, L., Cheng, Z., Li, K., Khan, S. U., & Li, K. (2016). An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment. *Journal of Grid Computing, 14*(1), 55–74.
12. Wang, L., Von Laszewski, G., Dayal, J., & Wang, F. (2010). Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS. In *Proceedings of the 10th IEEE/ACM international conference on cluster, cloud and grid computing*, pp. 368–377.
13. Durillo, J. J., Fard, H. M., & Prodan, R. (2012). Moheft: A multi-objective list-based method for workflow scheduling. In *Proceedings of the 4th international conference in cloud computing technology and science (CloudCom)*, pp. 185–192.
14. Yassa, S., Chelouah, R., Kadima, H., & Granado, B. (2013). Multiobjective approach for energy-aware workflow scheduling in cloud computing environments. *The Scientific World Journal, 2013*, 1–1138.
15. Zhu, Z., Zhang, G., Li, M., & Liu, X. (2016). Evolutionary multi-objective workflow scheduling in cloud. *IEEE Transactions On Parallel And Distributed Systems, 27*(5), 1344–1357.
16. Reyes, S. M., & Coello, C. C. (2006). ‘Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International journal of computational intelligence research, 2*(3), 287–308.
17. Coello, C. A. C. (2011) An introduction to multi-objective particle swarm optimizers. In *Soft computing in industrial applications*, pp. 3–12.
18. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Network, 1000*, 1942–1948.
19. Li, X. (2003) A non-dominated sorting particle swarm optimizer for multi-objective optimization. In *Proceedings of the genetic and evolutionary computation (GECCO)*, Springer pp. 198–198.
20. Fernandez, N., Alfonso, H. & Gallard, R.H. (2000). Crowding under diverse distance criteria for niche formation in multimodal optimization. *Journal of Computer Science & Technology, 1*(3).
21. Subashini, G., & Bhuvaneswari, M. C. (2011). 'Non dominated particle swarm optimization for scheduling independent tasks on heterogeneous distributed environments. *International Journal of Advance Soft Computing Application, 3*(1), 1–17.

22. Wakil, K., Badfar, A., Dehghani, P., Shoja Sadati, S. M., & Jafari Navimipour, N. (2019). A fuzzy logic-based method for solving the scheduling problem in the cloud environments using a non-dominated sorted algorithm. *Concurrency and Computation: Practice and Experience, 31*(17), e5185.

23. Garg, R. & Singh, A. K. (2011). Multi-objective workflow grid scheduling based on discrete particle swarm optimization. In *Proceedings of the international conference on swarm, evolutionary, and memetic computing*, Springer, pp. 183–190.

24. Beloglazov, A., Buyya, R., Lee, Y., & Zomaya, C. (2011). A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in computers, 82*(2), 47–111.

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

---

**Prathibha Soma** received the B.E. degree in Computer Science and Engineering from Gulbarga University in 1997, M.E. degree in Computer Science and Engineering from Sathyabama University in 2008 and Ph.D. from Anna University in 2019. Her research interests include Cloud Computing and Computer Networks.

**B. Latha** received the B.E. degree in Computer Science and Engineering from Annamalai University in 1998, Master’s in Computer Science and Engineering from Sathyabama University in 2005 and Ph.D. in the Faculty of Information and Communication Engineering, Anna University in 2010. Her current research interest includes Artificial Intelligence, Network Security, Machining of composite materials and optimization.
V. Vijaykumar is a EAI Fellow, Adjunct professor at School of Computer Science and Engineering, The university of New South Wales, Sydney, Australia. He has been serving as Editor and Associate editor for high impact journals. His research interest includes Cloud Computing, Artificial Intelligence and Fuzzy Systems.