An Improved Multi-Objective Workflow Scheduling Using NSPSO With Fuzzy Rules

SOMA PRATHIBHA (somaprathi25@gmail.com)
Sri Sai Ram Engineering College https://orcid.org/0000-0002-8669-6904

B. Latha
Sri Sai Ram Engineering College

V. Vijaykumar
University of New South Wales - Kensington Campus: University of New South Wales

Research Article

Keywords: Scientific Workflows, Cloud Computing, Fuzzy Rules,Particle Swarm Optimization, Energy efficiency and makespan

Posted Date: March 24th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-208053/v1

AN IMPROVED MULTI-OBJECTIVE WORKFLOW SCHEDULING USING NSPSO WITH FUZZY RULES

Soma Prathibha[1], B.Latha[2] & V.Vijaykumar[3]

[1]Associate Professor, Department of Information Technology, Sri Sai Ram Engineering College

[2]Professor and Head,Department ofComputer Science and Engineering ,Sri Sai Ram Engineering College

[3]Adjunct Professor,School of Computer Science and Engineering,The University of New South Wales, Sydney, Australia

Abstract

Lot of scientific problems in various domains from modelling sky as mosaics to understand Genome sequencing in biological applications are modelled as workflows with large number of interconnected tasks. Particle Swarm Optimization (PSO) based metaheuristics are currently used to address many optimization problems as they are simple to implement and able to produce quickly optimal or sub-optimal solutions based on learning capabilities. 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 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. An average of above 15% in the energy reduction for the proposed system over simple DVFS was achieved for all types of workflow applications with different dimensions. Similarly when compared to NSPSO an energy reduction of at least 10% has been observed for F-NSPSO for all three types of workflow applications. Compared to NSPSO

algorithm F-NSPSO algorithm shows at least 13% ,12% and 21% improvement in average makespan for Montage, Cybershake and Epigenomics workflow applications respectively.

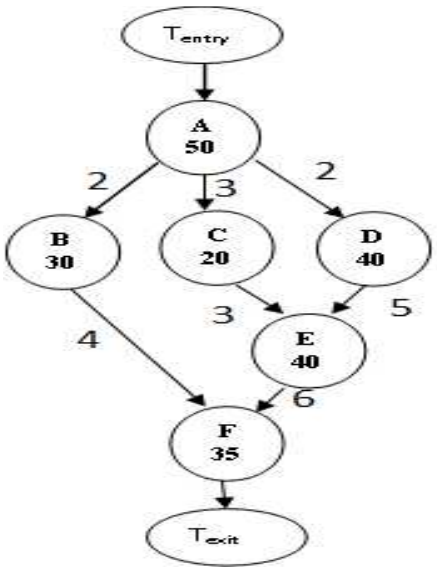
Keywords: Scientific Workflows, Cloud Computing, Fuzzy Rules,Particle Swarm Optimization, Energy efficiency and makespan

I. Introduction

Cloud computing has revolutionized the Information and Communication Technology (ICT) field by proving dynamic and highly scalable resources on-demand to clients on a pay as per resource usage. 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. Different fields of computing like grid, distributed cluster, and cloud aim at providing the computational power as a utility to many end users. Lot of scientific problems in various domains from modelling sky as mosaics to understand Genome sequencing in biological applications are modelled as workflows with large number of interconnected tasks. Scientific community is showing increasing interest in adopting the cloud platform for deploying workflow applications in because of its attractive features like dynamic resource provisioning, heterogeneous resources, pay for usage of resource and flexible billing models.In a white paper by Delforge & Whitney(2014),the authors have reported that energy consumption by IT equipments is very high and they contribute to overall 40 % of the energy consumption. In another survey(Sareh,2016) it is reported that cloud data centers contribute to 3% of the global energy consumption and it is expected to rise in future. The major resources for energy consumption in cloud data centers are servers,networking devices, memory, storage devices and other cooling equipments. Also, improper use of resources by the application also contributes to the high energy consumption by the cloud data centers. High energy consumption by the data centers will lead to the high maintenance cost for the cloud service providers, hence for the cloud service providers it is very important to minimize the energy consumption and at the same time satisfy the SLA parameters such as deadline and budget constraints of the users.

A.Features of Scientific workflow

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 (DAG) with vertices representing tasks of the application and edges depict the dependency between the tasks. Figure 1.1a shows DAG representation for an example workflow application. The numeric value associated with each vertex is the task execution time. Numeric value associated with an edge is the size of data transfer between two tasks. Starting and end of a graph is indicated with two special tasks T_{entry} and T_{exit} are inserted as pseudo tasks in the workflow.



	A	B	C	D	E	F
A	0	2	3	2	0	0
B	0	0	0	0	0	4
C	0	0	0	0	3	0
D	0	0	0	0	5	0
E	0	0	0	0	0	6
F	0	0	0	0	0	0

Figure 1.1 (a) and (b)DAG and matrix representation of example workflow application

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. Figure 1.1b shows the matrix representation for the example workflow application given in Figure 1.1a.

II. Literature Survey

Efficient scheduling of workflows is with advanced optimization techniques was done by applying new methods using Direct Acyclic Graphs (DAGs) which perform the scheduling operations in parallel using distributed systems. In general for any application execution resource management and scheduling in cloud platform is very complex and many authors have carried out their research work in this area. Figure 2.1 shows the Google trends graph for the research interest on the search term "Task scheduling in cloud computing"

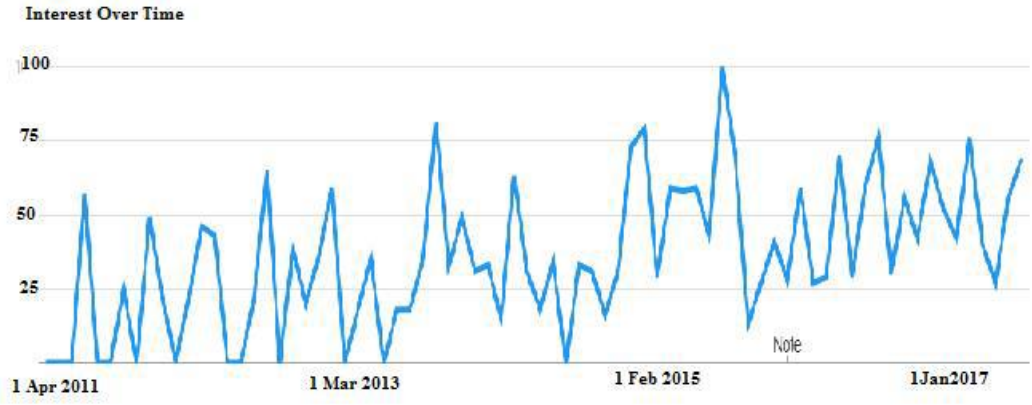


Figure 2.1: Google Trends Graph for search term Task Scheduling in Cloud Computing

Well established methods for optimizing workflows in distributed heterogeneous systems are available (Yu et al.,2008; Dong & Selim,2006) however they cannot be directly ported to cloud because of the unique characteristics of cloud like dynamic provisioning and innovative pricing models (Abrishami et al., 2013). These features add additional dimensions to the scientific computing in cloud platform. Further, scheduling jobs in a cloud resource is more complex than that of grid scheduling and involves additional variables in Service Level Agreements (SLA), policies that have to be maintained between the cloud resource provider and the user. Though cloud uses pay-per-use model to charge the customers on their resource usage, it rounds the resource usage to the next hour. This will make the customers pay for resources that were not actually, consumed by them. These criteria make cloud usage for scientific workload a little more complicated.

Several research works have been carried out for improving energy efficiency in cloud platform. This problem has been addressed by using techniques like task consolidation (Ferdaus *et al.* 2014), virtual machine consolidation (Lee *et al.* 2012;Mohanapriya *et al.*2018) have addressed energy efficiency issue by choosing VM with minimal energy consumption and also the authors use VM consolidation with double threshold scheme. Some works use heuristics based optimization towards minimizing both cost and energy consumption (Li.Z *et al.*2015; Pietri & Sakellariou 2014;Tang *et al.*2016) have proposed Dynamic Voltage Frequency Scaling(DVFS) based heuristic to minimize energy consumption in which the algorithm first finds the total makespan of running the application using HEFT. Then the processors are sorted based on their utilization. And all the underutilized processors are shut down and then the tasks are redistributed in the remaining processors and finally DVFS is applied to run the tasks using the low

frequency in the idle slots. Yassa *et al.*(2013a) have proposed multi-objective discrete particle swarm optimization to minimize the conflicting objectives energy consumption, makespan and cost. The authors use DVFS technique to control the processor voltage level for running the task at various voltage levels. Based on the requirement, the host machine voltage level supply is changed dynamically. Most of the above mentioned works use DVFS technique to adjust the frequency and voltage of the resources to minimize the energy consumption (Wang *et al.* 2010). However, reduction in voltage may not always ensure reduction in energy consumption, because running the application with less frequency may lead to the increase in the makespan of the workflow and hence it may result in more energy consumption. Durillo *et al.*(2012) use multi-objective list based optimization to improve the energy consumption and makespan for workflow scheduling in cloud.

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. The fitness function in the proposed work considers makespan, energy consumption and paid idle time of the resources. Also, there are multiple solutions which are good in different aspects, so in order to decide quickly the resource, task pair which is suitable for given user preferences fuzzy based decision selection is included in the NSPSO fitness function.

Cloud platform comprises of different types of resources with different billing options. To perform efficient workflow scheduling, all the resource and task combinations have to be checked which results in large solution search space. In order to satisfy both cloud provider and scientific user requirements, a high performing multi-objective optimization algorithm which generates all the quality solutions is required. As the set of quality

solutions is huge for large scale applications, hence an efficient fuzzy mechanism is needed which analyses these solutions quickly based on the users requirement.

3.Background

In this section basic concepts about multi-objective optimization is presented.Currently to solve many practical problems metaheuristic techniques have become more popular .These algorithms promise to be less cost and easy to understand and implement many real time applications which have multi-objective constraint. [13]

Traditional approach for solving multi-objective optimization was done by converting the problem into single objective optimization and it is known apriori approach. Weighted sum approach is one of the popular algorithm from apriori methods. Advantage of weighted sum approach is that it is simple and less calculations are needed to get the solution. However, the main drawback is that weighted approach provides only a single solution and hence trade-off analysis cannot be performed. 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 draw backs, most of the multi-objective optimization problems are solved using Pareto-optimal set which is generated by using Pareto-dominance relation. Pareto dominance relation is used for determining which optimal solution is better. In Pareto dominance relation instead of obtaining a single optimal solution, it emphasizes finding a set of alternatives with different trade-off values among the objectives. These solutions are called Pareto optimal solutions or non-dominated solutions.

3.1. Basic concepts of multi-objective optimization :

Most of the engineering problems currently use Multi-objective optimization which refers to the optimization with more than one objective (criterion) function .

This is modelled as [30,31]:

Definition 1:Multi-objective optimization

$$\text{Minimize : } X(z) = [x1(z), x2(z), \ldots, xn(z)] \quad (3.1)$$

$$\text{Subject to: } yi(z) \geq 0,\ i = 1, 2, \ldots, m \quad (3.2)$$

$$wi(z) = 0,\ i = 1, 2, \ldots, p \quad (3.3)$$

where $z = [z1, z2, \ldots, zk]$,T is the vector of decision variables, m is the number of inequality constraints, p is the number of equality constraints, y_i is the i^{th} inequality constraints, w_i is the i^{th} equality constraints, and obj is the number of objective

functions fi : Robj → R.

The major challenge in multi-objective optimization is relational operators cannot be used for comparing the solutions in the search space and special operators called pareto dominance is used for comparing the solutions in the search space of multi-objective optimization problem[16,17,18]

3.1.1 Pareto Optimal Solutions

Pareto optimal set denotes the notion of solution dominance in multi-objective optimization problems. A solution X is considered to dominate another solution Y, if X is better than Y in at least one of the objectives and better than or equal to in other objectives.

Definition 2:Pareto-Optimal Set

Formally, Pareto optimality is defined as, a solution x $\alpha \in X$ is Pareto optimal if there does not exist another solution x$\in X$such that$f(x) \prec paretof(x^{\alpha})$

Pareto optimal set is defined as follows

$$P^* = \{x \in \mathrm{X} \mid \nexists y \in X: f(y) \preccurlyeq f(x)\} \quad (3.4)$$

Definition3:Pareto-front

Pareto front is set of all Pareto optimal points defined as

$$PF^* = \{f(x) = \{f1(x), \ldots . fk(x) | \mathrm{x} \in P^*\}\} \quad (3.5)$$

3.1.2 Non-Dominated Sorting PSO

The main goal of multi-objective optimization problem is finding as many solutions as possible which are close to the Pareto-front. Figure 3.1, shows an example for Non-dominated Sorting technique in which three fronts, namely first front, second front and third front are considered.

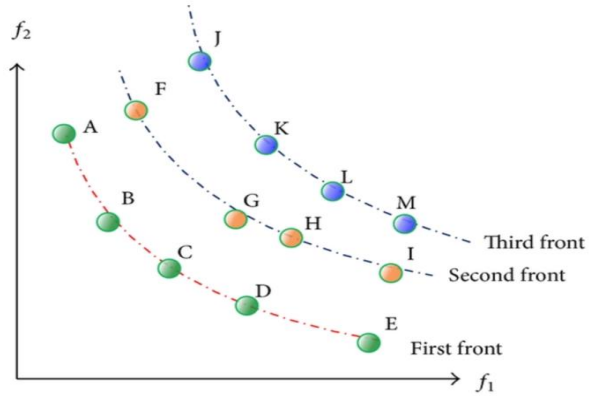


Figure 3.1 Example for Non-dominated Sorting

Basic PSO cannot be used for identifying such solutions as it is designed to obtain single global best solution. MOPSO introduces a large number of techniques for creating selection pressure on particles to move towards true Pareto front. One such technique called the Non-dominated sorting PSO is adapted in this work. Basic PSO is extended in NSPSO by incorporating effective comparison between each particle's local best and its offspring. NSPSO method overcomes the drawback of the basic PSO by increasing the resource sharing among particles and its offspring (Li 2003). Here, N initial particles are combined with its offspring 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. Unlike basic PSO, NSPSO compares all particles' personal bests and offspring of all other particles in the entire population to provide an appropriate selection procedure to push the swarm particles towards the true Pareto optimal front. This helps to identify a large number of non-dominating solutions.

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

Let P_1^t and P_2^t represent two particles considered in PSO algorithm at time instant t and $F(P_1^t)$ and $F(P_2^t)$ represent the evaluation of fitness values for particles P1 and P2. $F(X_1^{t+1})$ and $F(X_2^{t+1})$ represent fitness values for P1 and P2 at time instant t+1. In basic PSO, the comparison is performed between particle and its offspring. For example in this scenario, the basic PSO makes the comparison between only $F(P_1^t)$ and its offspring $F(X_1^{t+1})$, similarly between $F(P_2^t)$ and its offspring $F(X_2^{t+1})$. This results in losing some of the valuable non-dominated solutions. This drawback is overcome in NSPSO by comparing each particle with all the offspring resulting in comparison of 2N particles. Considering the above example, if NSPSO is applied, then comparison is done between all the four particles and it helps in retaining useful non-dominating solutions.

NSPSO uses the concept of non-dominated Sorting from NSGA-II to sort the entire population into different non-dominating levels. Non-dominated sorting arranges the population such that the first front consists of population which are non-dominated solutions and the second front contains solutions which 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 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 Equation (5.3) for a lower limit l, upper limit u and a value m which lies between a and b.

$$\mu_A(x) = \begin{cases} 0 & x \le l \\ \dfrac{x-l,}{m-l} & l < x \le m \\ \dfrac{u-x}{u-m} & m < x < u \\ 0, & x \ge b \end{cases} \quad (3.6)$$

The basic PSO produces fast convergence which reduces the diversity of the swarm. In NSPSO, 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 population.

Niche Count Calculation: Niche count of a particle Pi indicates closeness of the particle to its neighbourhood (Fernandez et al. 2000). It is calculated dynamically using Euclidean distance as given in Equation (3.7)

$$\sigma_{\text{share}} = \frac{u_2 - l_2 + u_1 - l_1}{N-1} \quad (3.7)$$

In the Equation (3.7), ui and li 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 as a good solution. The existing NSPSO algorithm uses the computation of Niche count for providing better solutions.

4.PROPOSED SYSTEM

A new Multi-Objective Optimization model using F-NSPSO Meta-Heuristics is proposed in this work. 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 the scientists to choose suitable resource configurations from the set of available solutions. Therefore, a novel multi- objective optimization techniques has been proposed in this work using Non-dominating sorting operations and Particle Swarm optimization which enhances the quality of schedules by comparing all the possible solutions

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. The fitness function in the proposed work considers makespan and energy consumption Also, there are multiple solutions which are good in different aspects, so in order to decide quickly the resource, task pair which is suitable for given user preferences fuzzy based decision selection is included in the NSPSO fitness function.

NSPSO algorithm has been used by Subashini et al.(2011) to minimize flow time and makespan for scheduling set of independent tasks in 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 problem in cloud platform. Garg& Singh (2011) have used NSPSO meta- heuristic for workflow scheduling in grid platform. The authors in their work have used NSPSO to generate Pareto-fronts with minimization objective of makespan and cost. However, the proposed work differs from the above mentioned work by considering all the characteristics of the resources in the cloud platform 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 [WT_1,WT2,...,WT_n] to resources[R_1,R_2,...,R_m] 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 this work, solution for workflow scheduling with an objective of minimizing energy consumption and makespan is proposed using highly efficient Fuzzy and Pareto based Metaheuristics called Non-dominated Sorting Particle Swarm Optimization (F-NSPSO). 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 the conflicting objectives, one of the requirements for the scientists is that they would be interested in getting an optimal cloud configuration with 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 cloud platform is 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

$$E_{total}(G)=E_{computations}(G)+E_{datatransfer}(G) \quad (4.1)$$

In the above equation $E_{computations}$ (G) represents 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 $E_{computations}$ (G) represents 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 VM is running .In the proposed work, energy consumption for the data transfer

is not addressed. Power consumption of a processor in VM consists of static and dynamic consumption (Beloglazov *et al.* 2011). As the static consumption is not of much significance, we consider only the dynamic part. Dynamic power consumption is computed using the following formula.

$$P_{dynamic} = A \times C \times V_{jl} \times f^2 \quad (4.2)$$

where A is the number of switches per clock cycle, C is the total capacitance load, V_{jl} is the supply voltage at level l on the processor j, and f is the operating frequency that operates the supply voltage at level l. The parameters A and C, which are device related constants, depend on each device capacity. The parameters v and f are proportional to the computation capacity of the processor and they operate in the range of $[v_{max}, v_{min}]$ and $[f_{max}, f_{min}]$ respectively. Thus the energy consumption of a task t_i during its runtime is calculated as follows

$$E_{dynamic} = P_{dynamic} \times t_{i_{exe—time}} \quad (4.3)$$

Above equation indicates that the energy consumed by a processor of a physical machine on which VM is running and it is computed as a product of power consumption and the total time required to execute the task ti . When the processor is in an idle state, it operates at vmin and fmin and the power consumption is computed using the following equation.

$$P_{idle} = A\ X\ C\ X\ V_{jmin}\ x\ f_{min}^2 \quad (4.4)$$

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

$$E_{idle} = P_{idle} \times t_{idle\_time} \quad (4.5)$$

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

$$E_{total}(VM) = E_{dynamic} + E_{idle} \quad (4.6)$$

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

$$E_{computations}(G) = \sum_{i=1}^{n} E(VM_i) \quad (4.7)$$

4.1.2Workflow Makespan Calculation

One more important performance measure for scientific workflow application is makespan which indicates the overall execution time of the workflow. Since workflows represents interdependent tasks ,hence the makespan model is developed using set of recurrence equations. $WF\_Task_{start\_time}(wt_i, r_j)$ is calculated using equation 3.8 by choosing the maximum time between available task and ready task.Time for completing ready task depends on finish time and communication cost of task as shown in equation 3.9

$$WF\_Task_{start\_time}(wt_i, r_j) = \begin{cases} 0 \quad \text{for } wt_i = wt_{entry} \\ \max\{ WF\_TASK_{avail}(r_j), WT_{ready}(t_i, r_j)\} \text{for } t_i \neq t_{entry} \end{cases} \quad .....(4.8)$$

$$WF\_TASK_{ready}(wt_i, r_j) = \max\{WF\_TASK_{finish_{time}}(wt_k) + comm\_cost(e_{k,i}) \quad (4.9)$$

where $wt_k \in Pred(wt_i)$,

$WF\_TASK_{finish\_time}(wt_i, r_j)$ represents the end time of task wt_i on resource r_j which is calculated using Equation (3.10).

$$WF\_TASK_{finish\_time}(wt_i, r_j) = WF\_TASK_{start\_time}(wt_i, r_j) + WF\_TASK_{exe\_time}(wt_i, r_j) + Datatransfer\_time(wt_i, r_j) \quad (4.10)$$

Makespan of the workflow application is obtained by finding finish time of task wt_{exit}

$$Makespan(G) = WF\_TASK_{finish-time}(wt_{exit}) \quad (4.11)$$

4.2 PARETO-BASED WORKFLOW SCHEDULING USING F- NSPSO

In the proposed work, non-dominated solutions generated by 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. The resultant Pareto-fronts has been used in this work to provide an optimal configuration of cloud to the scientific users. Moreover, 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 on various scientific workflow applications with different 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.

Step 1: Apply **t**opological sorting algorithm to maintain task dependencies of the workflows.

Step 2: Initialize the Virtual machine array VM [Id, VM Type, Available_ time]

Step 3: Map the tasks in the sorted result to various instances for generating different particles

Table 4.1 shows the sample particle encoding used in F-NSPSO for the tasks of example workflow application shown in figure 4.1 and VM details given Table 4.2

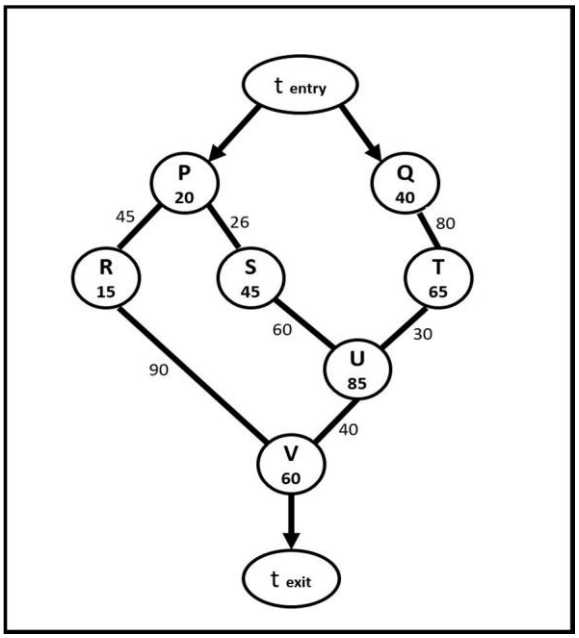


Figure 4.1 Example workflow application used to illustrate F-NSPSO

Table 4.1 Description of Virtual Machine Array

ID	Type	Available_time
VM1	Small	1 Hour
VM2	Medium	50 minutes
VM3	Small	3 Hour
VM4	Large	2 Hour

Table 4.2 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

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.

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-Numberofparticles, 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_iterationsdo
10    for each  particle p ∈PSOListdo
11 SelectGbest for p from a specified high part    (5%)
12       calculate v_{t+1}  andp_{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_populationdo
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 shows the computation of fitness values using the fitness function shown in Equation 3.12

$$Bt\_Val = \frac{(W1 \times p.\ makespan + W2 \times p.\ Power\_consumed + W3 \times p.\ Paid\_idle\_time)}{(W1 + W2 + W3)} \quad (4.12)$$

where, the best value is the fitness function value, computed using the weights W1=0.5, W2=0.3 and W3=0.2. The values for the weights W1, W2 and W3 were estimated through repeated experiments for finding the optimal values.

Algorithm 4.2 Fitness Function Computing in F-NSPSO

```
1   Function Fitness_Function(PSO_List)
2   for every ready task t ∈ T do
3     find t.length, t.memory,t.input_data,t.output_data
4   end for
5   for each virtual machine ∈ R do
6     find vm.cpu_cores, vm.memory, vm.storage,
    vm.bandwidth, vm.mips,vm.available_time
7   end for
8   for each particle p ∈ S do
9   Calculatep.makespan
10  Calculatep.energy_consumption
11  Calculatep.paid_idle_time
12        Find Best_Value using equation 5.13
13  if Best_Value>High_Value then
    Fitness_Value = Best_Value
    else if Best_Value>Medium_Value then
    Fitness_Value = 2*Best_Value
    else // if Best_Value>Low_Value
```

Fitness_Value = 3*Best_Value **end**
if
14 **end for**
15 **return Fitness_value**
16 end procedure

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 systems applies rules which are stored in a knowledge base against facts present in the application to perform inference. Table 3.3 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 the table 4.3 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 the Table 4.3 represents the rule:

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

In this way, the fuzzy inference system developed in this work applies the fuzzy rules using forward chaining inference mechanism to perform deductive inference in order to make efficient decisions on configurations and the scheduling methods. In memory size, Low memory size indicates a memory size up to 32 GB. Medium size of memory indicates a memory upto 1 TB. Finally, High memory indicates the availability of memory upto 1PB through virtual machine allocation. In the case of Energy Consumption, Less energy indicates the energy upto 1KW. Average energy consumption indicates upto 1000 KW and High energy consumption indicates upto 100 MW. In the case of

Time to Complete the Workflow Task, small indicates the time to complete the task upto 1 hour, Medium indicates the time to complete the task upto 2 hours, High indicates the time to complete the task upto 3 hours. Hence, the proposed scheduling algorithm aims at optimizing these parameters in the cloud data center by applying the proposed method.

Table 4.3 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	High	Small	Good
Medium	Average	Large	Very Fair
Medium	Average	Medium	Good
Medium	Average	Small	Very Good
Medium	Less	Large	Good
Medium	Less	Medium	Very Good
Medium	Less	Small	Best
High	High	Large	Very Fair
High	High	Medium	Good
High	High	Small	Very 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

5. Results and Discussion

In this section, the description of parameter settings of this work for implementing the proposed F-NSPSO and set of real world workflow application used in the experiment are presented. Table 5.1 shows the parameters used in the experiments.

Table 5.1 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 5.2 lists the characteristics of various workflow applications used in the experiment.

Table 5.2 Characteristics of Real world workflow application

Workflow	Type	Number of Levels	Number of Tasks	Number of Edges	Average Execution Time ofTasks (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

5.1 Experiment 1: Makespan analysis

To verify the performance of the proposed algorithm, initially the results of makespan and energy consumption are compared with HEFT and DVFS algorithm. In DVFS based implementation, the jobs are assigned to the machines with minimum energy consumption by operating the processor at low voltage.

Figures 5.3 and 5.4 show the performance of the proposed F- NSPSO with the existing task scheduling algorithms for Cybershake, and Montage applications. Here, five experiments have been conducted with five different numbers of tasks such as 100, 200, 300, 400 and 500. Figure 5.5 shows the performance F-NSPSO for Epigenomics workflow applications with task set 10,20,30,40 and 50. The makespan values are considered for measuring the performance of the various experiments.

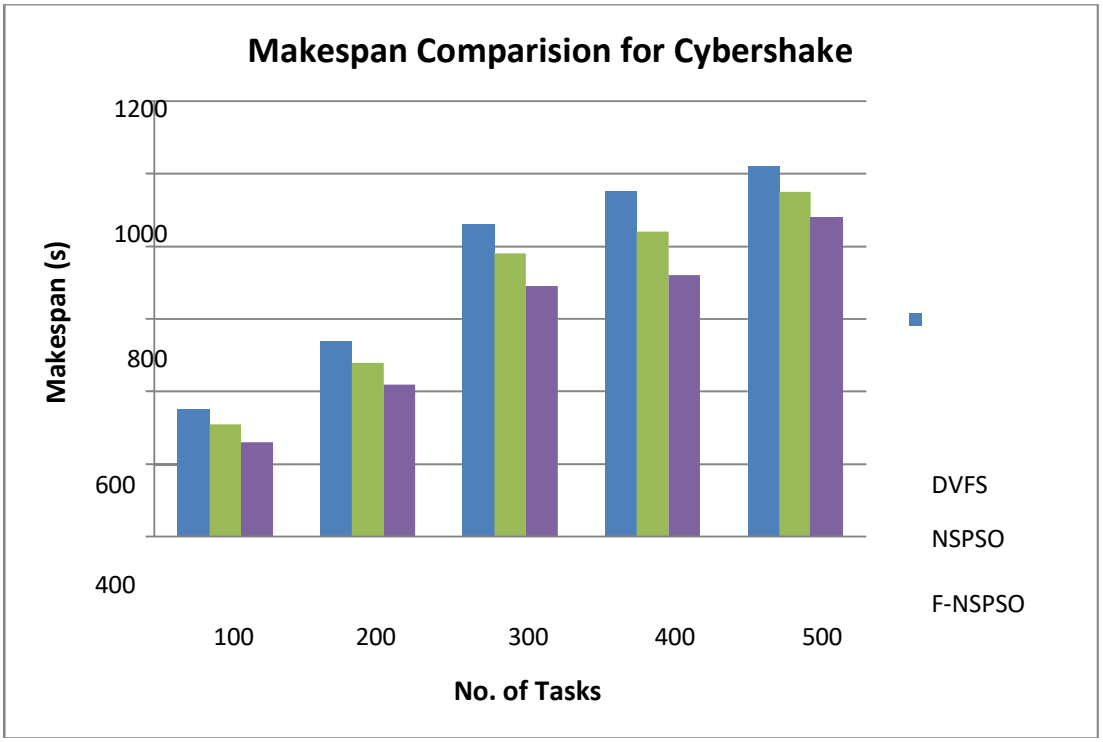


Figure 5.3 Comparison of Makespan value for Cybershake with F-NSPSO

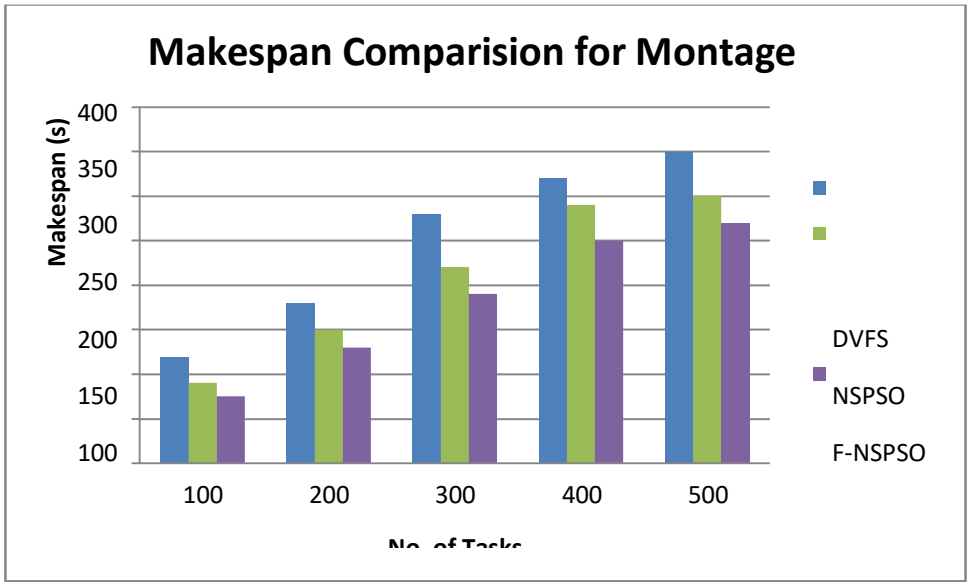


Figure 5.4 Comparison of makespan value for Montage with F-NSPSO

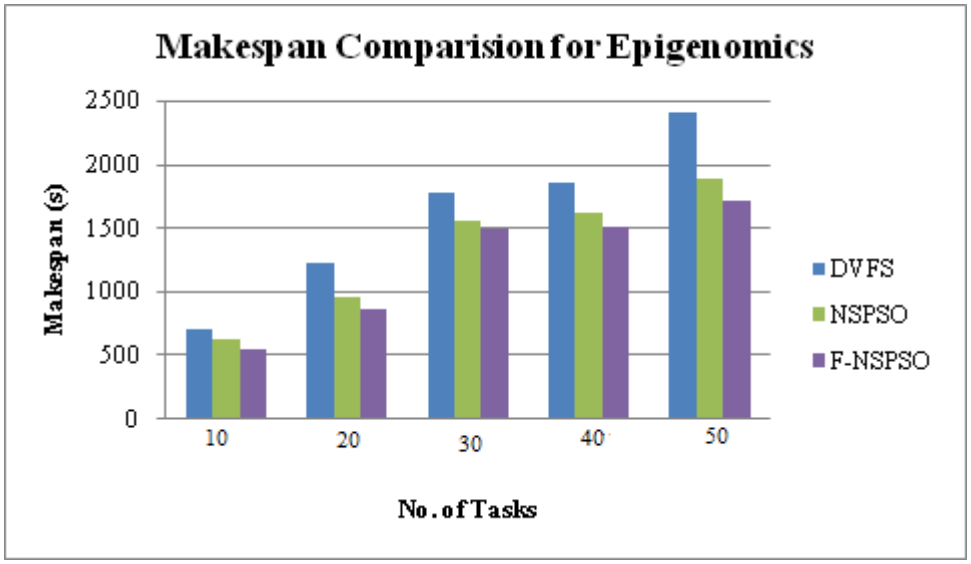


Figure 5.5 Comparison of makespan value for Epigenomics with F-NSPSO

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, HEFT and NSPSO in terms of makespan values. Compared to 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 the three applications over DVFS algorithm.

Experiment 2: Comparison of Energy Consumption

Experiments were conducted to evaluate energy consumption performance of F-NSPSO with DVFS and NSPSO .The performance on The Y-axis represents the gain obtained for each workflow which is calculated using equation 4.1 and the X-axis in the figure represents the particle dimension for each workflow.

$$\text{Energy Reduction} = \frac{\text{TotalEnergy}_{\text{DVFS or NSPSO}} - \text{TotalEnergy}_{\text{F—NSPSO}}}{\text{TotalEnergy}_{\text{DVFS or NSPSO}}} \quad (5.1)$$

An average of above 15% in the energy reduction for the proposed system over simple DVFS was achieved for all types of workflow applications with different dimensions. Even though simple DVFS is an efficient energy saving mechanism, the proposed system is performing better because it looks for all the combinations of resources that can exist in the dynamic cloud environment. Similarly when compared to NSPSO an energy reduction of at least 10% has been observed for F-NSPSO for all three types of workflow applications.

Figures 5.2,5.3 and 5.4 show the energy consumption analysis between the proposed task scheduling algorithm called F-NSPSO and the existing scheduling algorithms such as DVFS and NSPSO for Epigenomics,Montage and Cybershake. Here, we have conducted five experiments by considering the various numbers of tasks for each application.

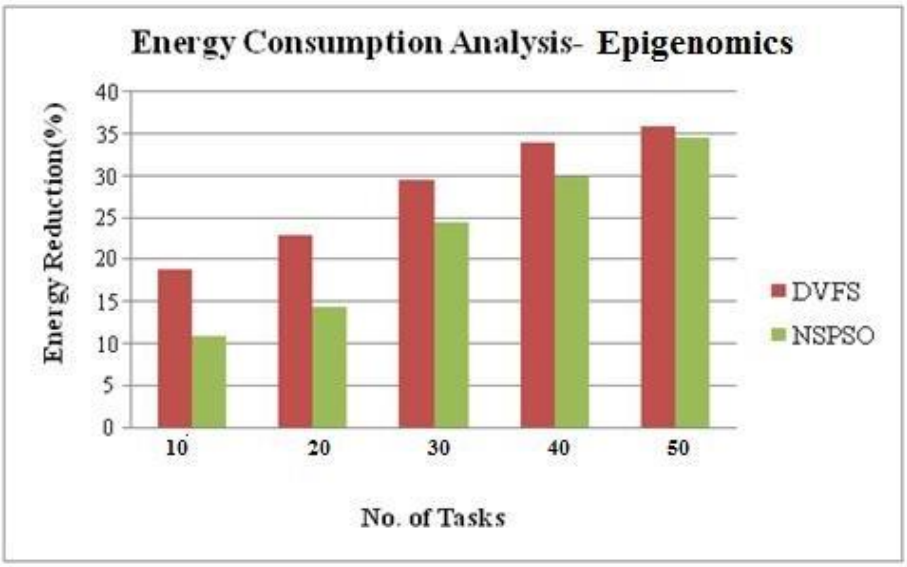


Figure 5.2 Energy Consumption analysis for Epigenomics Workflow with F-NSPSO

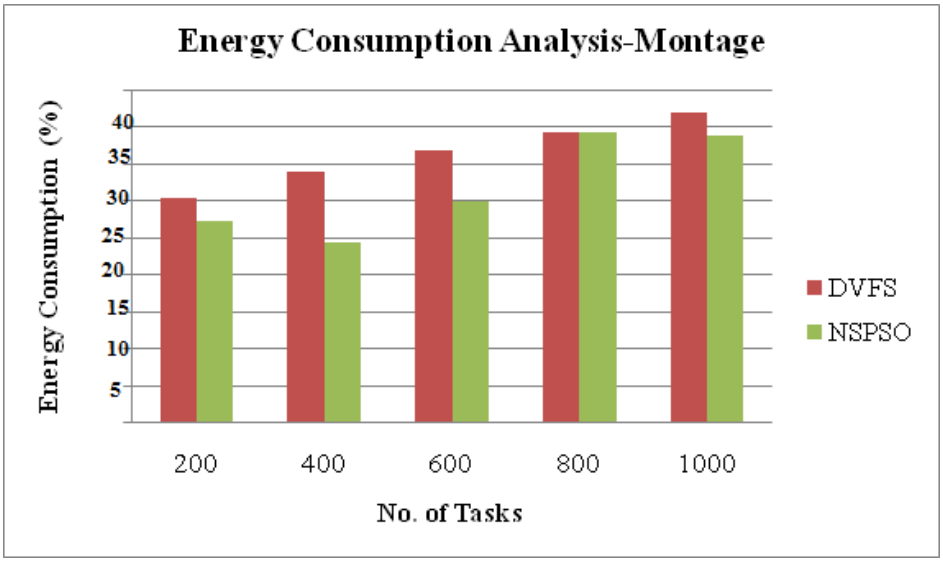


Figure 5.3 Energy Consumption analysisfor Montage Workflow with F-NSPSO

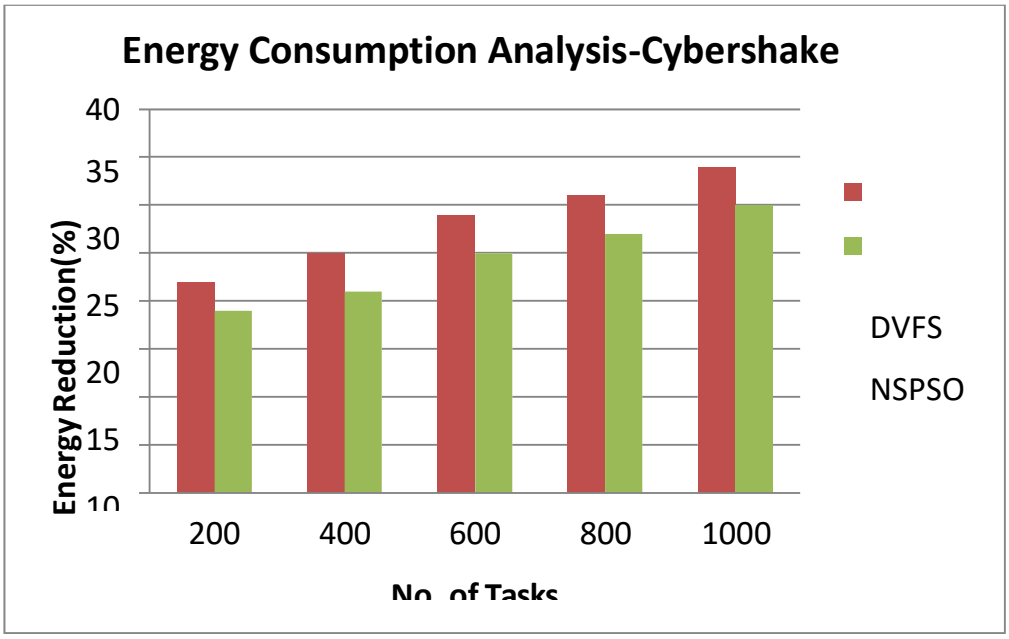


Figure 5.4 Energy Consumption analysis for Cybershake Workflow with F-NSPSO

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. This is due to the use of effective optimization process which is carried out by incorporating the appropriate fuzzy membership function

6.Conclusion

In this work a pareto based solution for workflow scheduling using multi-objective optimization based on Fuzzy-NSPSO has been developed. The algorithm uses a fitness function to minimize the energy consumption, makespan and paid idle time of the resources. For large scale scientific applications number of non-dominated solutions will be more ,for deciding the quality of the solution memory utilization is also considered as

one more 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. Simulations have been conducted to evaluate the performance of F-NSPSO on three real world scientific applications Epigenomics, Cybershake and Montage. F-NSPSO has given better performance compared to HEFT,DVFS and NSPSO. Performance analysis of F-NSPSO algorithm shows that the quality of the non-dominated solutions is good.

Declaration

- **Funding** (information that explains whether and by whom the research was supported)-Not Appliocable
- **Conflicts of interest/Competing interests** (include appropriate disclosures):No conflict of interest/competing interest
- **Availability of data and material** (data transparency)-Yes
- **Code availability** (software application or custom code)-Yes
- *Additional declarations for articles in life science journals that report the results of studies involving humans and/or animals*
- **Ethics approval** (include appropriate approvals or waivers)-NA
- **Consent to participate** (include appropriate statements)-NA
- **Consent for publication** (include appropriate statements)-NA

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,FP 2016,'Energy-Efficient Management of Resources in Enterprise and Container based Clouds', The University of Melbourne.

3. Dong, F. & Selim, 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 In Computational Intelligence, vol. 146, pp. 173-214

5.Abrishami, S, Naghibzadeh, M & Epema, DH, 2013. 'Deadline constrained orkflow scheduling algorithms for infrastructure as a service clouds'. Future Generation Computer Systems, vol. 29, no. 1,
pp. 158-169.

6. Ferdaus, MH, Murshed, MM, Calheiros, RN & Buyya, R 2014,'Virtual Machine Consolidation in Cloud Data Centers Using ACO Metaheuristic', Proceedings of European Conference on Parallel Processing, Springer, Cham, pp. 306-317.
7. Lee, YC & Zomaya, AY 2012, 'Energy efficient utilization of resources in cloud computing systems', Journal of the Supercomputing, vol. 60, no. 2, pp. 268-280.
8. Mohanapriya, 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,vol.34,no.3, pp.1561-1572
9. Li,J, Li,YK,Chen,X, Lee,PP & Lou, W 2015, 'A Hybrid Cloud Approach for Secure Authorized Deduplication', IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 5, pp.1206-1216
10. Pietri, I & Sakellariou, R 2014, 'Cost-efficient provisioning of cloud resources priced by CPU frequency',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, SU & Li, K 2016, 'An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment', Journal of Grid Computing, vol.14,no.1, pp.55-74
12.Yassa, S, Chelouah, R, Kadima, H & Granado, B 2013a, 'Multiobjective approach for energy-aware workflow scheduling in cloud computing environments', The Scientific World Journal, vol. 2013,Article ID350934, pp. 1-1138.
13 Garg, R & Singh, AK 2011,'Multi-objective workflow grid scheduling based on discrete particle swarm optimization'.Proceedings of the International Conference on Swarm, Evolutionary, and Memetic Computing , Springer, Berlin,Heidelberg, pp. 183-190
14 Beloglazov, A, Buyya, R, Lee, Y, C & Zomaya, A 2011, 'A taxonomy and survey of energy-efficient data centers and cloud computing systems', Advances in computers, vol.82,no.2, pp.47-111.
15. G. Dhiman, V. Kumar, Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications, Adv. Eng. Software 114 (2017) 48–70, doi:10.1016/j.advengsoft.2017.05.014].

Figures

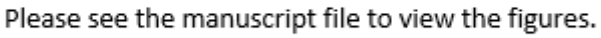


Figure 1

Please see the manuscript file to view the figures.