Cluster Computing (2024) 27:1965–1982
https://doi.org/10.1007/s10586-023-04057-z

User request-based scheduling algorithms by managing uncertainty of renewable energy

Slokashree Padhi[1] · R. B. V. Subramanyam[1]

Received: 25 June 2022 / Revised: 20 May 2023 / Accepted: 23 May 2023 / Published online: 15 June 2023

Abstract
The overwhelming growth of cloud computing has introduced new challenges for the cloud infrastructure. Many cloud service providers (CSPs) have been adopting renewable energy (RE) sources to increase profitability and reduce carbon emissions. Therefore, recent literature focuses on managing cloud infrastructure with RE sources in addition to traditional non-renewable energy (NRE) sources whenever required. However, these works aim to maximize the usage of RE or minimize the cost without considering the uncertainty (UN). This paper presents three user request-based scheduling algorithms, namely UN-based future-aware best fit, UN-based round-robin, and UN-based highest available renewable first, by managing the UN of RE and NRE. These algorithms consider two types of UN, namely UN-RE and UN-NRE, concerning the user request. The proposed algorithms undergo a rigorous simulation process with 200 to 2000 user requests and 20 to 200 datacenters and are compared using overall cost, UN cost and time, and the number of used RE resources.

Keywords Cloud computing · Renewable energy · Uncertainty · Non-renewable energy · Scheduling · User request · Datacenter

1 Introduction

The global market of cloud computing has a tremendous upward trajectory and transformed our way of living and livelihoods [1–4]. It also reforms the way of conducting business, developing applications, and configuring infrastructure [5]. As per Cloudwards [6], 94% of enterprises rely on various cloud services. These services include storage, server, database, application, and many more [7]. Therefore, the compound annual growth rate (CAGR) of the cloud computing market is expected to reach 17.5% and be worth around $832.1 billion by the end of 2025 [6]. More specifically, the CAGR of the cloud computing storage market from 2021 to 2028 is expected to reach 26.2% and be worth around $390 billion by the end of 2028 [6].

✉ Slokashree Padhi
slokashree.padhi@gmail.com

R. B. V. Subramanyam
rbvs66@nitw.ac.in

1 Department of Computer Science and Engineering, National Institute of Technology (NIT) Warangal, Warangal, Telangana 506004, India

The overwhelming growth of the cloud marketplace opens up new challenges to cloud service providers (CSPs). The challenges include infrastructure management, energy consumption, cost, security, customization, vendor lock-in, expanded service availability, and skilled engineers [8–14]. One of the biggest challenges is power management, as it directly impacts the cost, environment, and outages. For instance, the average power usage effectiveness (PUE) of large datacenters is approximately 1.55 as per the survey conducted in 2022 [15]. Most of the datacenters hosted by CSPs primarily depend on the electric grid. The uninterrupted supply of electricity will ensure the delivery of cloud services timely. Still, the CSPs maintain backup power systems to run their datacenters without any interruption. The primary source of electricity and backup power systems is fossil fuels (i.e., coal and its products, gas and its variants (natural and derived), petroleum, etc.) [16]. However, these sources are unsustainable and harmful to our physical environments, such as climate change, water pollution, and air pollution. Therefore, many CSPs have been planning to minimize their carbon footprints, water usage, electricity usage, and so on. One of the possible solutions is to adopt renewable energy (RE) sources, namely sunlight/solar, water/hydropower, wind, biomass, geothermal and tidal [17, 18]. For instance, Google claims that it is the largest buyer of renewable energy, and it has attained 100% renewable energy in its datacenters since 2017 [19]. Microsoft has also powered its datacenter using 100% renewable energy since 2014 [19]. Amazon has announced that it has achieved 85% renewable energy and is committed to attaining 100% by 2025 [19]. Moreover, these sources are sustainable and not harmful to the environment. Therefore, recent literature (especially, future-aware best fit (FABEF) [17], static cost-aware ordering (SCA) [20], round-robin (RR) [17] and highest available renewable first (HAREF) [17], MinBrown (MB) [21]) focuses on RE sources for managing the resources of datacenters without detaching the traditional way of powering datacenters using non-renewable energy (NRE) sources. However, these studies are limited to maximizing the usage of RE or minimizing the cost without emphasizing the uncertainty (UN). Here, UN refers to the delay in execution of user request (UR) due to internal and external factors [22]. Some examples of these factors include network failure, virtual machine interruption, checkpointing, overbooking, economics, load structure, human resources, power supply, air conditioning, and many more [23]. CSPs need to adopt a number of precautions for dealing with these UN factors. Otherwise, the UN leads to the violation of the service-level agreement (SLA) and customer dissatisfaction.

This paper addresses the problem of matching and scheduling URs to the resources of datacenters by managing the UN of RE and NRE resources. It presents three UR-based scheduling algorithms for cloud computing, namely UN-FABEF, UN-RR, and UN-HAREF. These algorithms take into consideration two types of UN, namely UN-RE and UN-NRE, and model the UN in terms of the percentage of UN-RE and UN-NRE. The UN-RE is variable, and the UN-NRE is fixed as RE and NRE resources are unreliable and reliable, respectively. For instance, consider a UR (i.e., UR_1) that requires one node for four time slots in one of the two available datacenters. The UN of this request is pre-determined as 50% for RE and 10% for NRE. The request is matched with datacenters DC_1 and DC_2. In both datacenters, four RE resource slots are available. Therefore, there is no need to assign the NRE resources. The UN is calculated by multiplying four RE resource slots and 50%, which is 2. This means that the duration of UR_1 can be extended from 4 to 6 (i.e., $4 + 2$) by incorporating the UN. Note that this is essential for fulfilling the requirements as per the SLA. UN-FABEF matches the UR to all the available datacenters and determines the assignment cost, including UN time duration. Then it assigns that UR to a datacenter that results in the least assignment cost. UN-RR assigns the URs to the datacenters in a circular fashion. On the contrary, UN-HAREF matches the UR to all the available datacenters and determines the number of RE resource slots, including UN time duration. Then it assigns that UR to a datacenter that results in the highest number of RE resource slots. The performance of the proposed algorithms is assessed through a rigorous simulation process by incorporating UN and tested with 200 to 2000 URs and 20 to 200 datacenters. The results are obtained in terms of overall cost, UN cost and time, and the number of used RE resources. Note that there is no dependency between the overall cost and renewable energy usage. To the best of our knowledge, existing literature does not consider the UN of URs in terms of the percentage of UN-RE and UN-NRE. Therefore, the proposed algorithms are compared among themselves to show their applicability.

The significant contributions of this paper are listed as follows.

1. Development of three UR-based scheduling algorithms by managing the UN of RE and NRE sources.
2. The UN of UR is modelled in terms of the percentage of UN-RE and UN-NRE and considered as a variable for RE resources and fixed for NRE resources as per their sustainability.
3. The UN time is determined using the number of RE and NRE resource slots that are assigned to the UR. This time can be considered as a reserved time to handle the UN.
4. Simulation of three proposed algorithms in ten different datasets and comparison of results in four different performance metrics.

The remaining sections of this paper are arranged as follows. Section 2 shows the related work on RE and NRE-based scheduling algorithms with their scope of improvements. This section also emphasizes UN factors for cloud computing. Section 3 presents the RE-based system model with UN and the problem statement with objectives and constraints. Section 4 focuses on three proposed algorithms, their pseudocode, illustrations, and complexity analysis. Section 5 discusses the performance metrics, the simulation configuration, the generation of datasets, the simulation results, and the performance analysis. Section 6 summarizes this paper with some further extensions.

2 Related work

The research on RE for sustainable development is rapidly increasing and is the need of the hour. The CSPs are willing to run their datacenters using RE sources to provide low-cost services and attract more enterprises to increase profitability. Still, they face many challenges with respect to supplying power to the datacenters. Therefore, researchers [17, 23–29] suggest using both RE and NRE sources to handle any future circumstances. However, the UN of these sources needs to be taken care of in order to handle the URs without violating their requirements.

Majid and Kumar [25] have discussed RE development in various states/countries/continents and presented predictions, employments, and obstacles. They have shown the projections from 1990 to 2040 by targeting some regions, namely Africa, Brazil, China, European Union, India, Middle East, Russia, and United States. They have reported that the lack of regulations and policies is one of the significant barriers to the adoption of RE. In the cloud context, the standardization of SLA is needed to avoid this barrier. Methenitis et al. [26] have adopted SLA by incorporating cost, reliability, and quantity. The rationality behind this adoption is that electricity production using RE sources is uncertain, and the delivery of the same cannot be promised. However, their study focuses on the smart grid and does not consider the UN of NRE sources. Ahmed [27] has reported that the traditional SLA does not cover the eco-friendly, green, and information technology ethics issues. As a circumstance, they have suggested a green SLA framework by incorporating these issues and adding a new layer to their framework. However, the UN is not considered in the green SLA framework. Carvalho et al. [29] have listed the factors influencing the energy regulatory process on the energy clouds. Specifically, they have considered twenty-nine factors grouped into seven types: availability, economic, ideology, information, institutional and market, personal and regulatory infrastructure. They have suggested establishing the mathematical model by analyzing the relationships between these factors. This model can determine the crucial factors that need immediate attention. Our paper highlights some of these types by developing UR-based scheduling algorithms.

Kabir et al. [23] have explored various uncertainties and their possible impact on the cloud components. They have identified five parameters affecting the UN: availability, cost, security, traffic, and workload. They have shown the impact of common influencing factors, such as user location, arrival time, checkpointing, execution time, virtual machine interruption, network failure, and closest datacenters, on the five parameters. Our paper focuses on two parameters, namely availability and cost, and considers some of the influencing factors: arrival time, execution time, and network failure. Koutsoyiannis [28] has suggested that UN modelling is an essential step in managing the RE. This modelling needs both structural measures and optimization techniques. Our paper intends to focus on optimizing the resources and their cost. Panda and Jana [22] have considered UN for the low quality of service (QoS) applications. They have categorized the UN into low QoS with high UN and low QoS with low UN. Here, high and low UN is determined based on the deviation from the expected completion time. However, their study is limited to task scheduling without considering RE resources. Panda and Jana [30] have presented an energy-efficient algorithm for a heterogeneous cloud environment. The algorithm comprises three steps: estimation, normalization and selection, and execution. Although energy is considerably reduced using this algorithm, they have not taken RE sources into consideration.

Xu and Buyya [24] have addressed the problem of huge energy consumption in the datacenters. They have suggested managing the loads among various clouds that are deployed in different time zones to minimize the usage of NRE and maximize the usage of RE resources. However, their model considers RE, power consumption, and carbon emissions without looking into UN in load shifting. Le et al. [20] have identified that the cost and ambient temperature of datacenters vary based on the location. As a result, they have introduced a load distribution policy to save the cost to a greater extent. They have also shown three baseline policies: RR, worst fit (WF), and SCA. Here, the worst fit assigns the URs to the highest available resources datacenter. SCA assigns the URs to the least cost datacenter. Chen et al. [21] have presented a scheduling algorithm, MB, to minimize the usage of NRE resources by managing the load among the datacenters. The algorithm is based on the availability of RE resources, fine-grained scheduling, and cooling power. However, they have not considered cost analysis. Toosi and Buyya [17] have developed a load-balancing algorithm using fuzzy logic (FLB). This algorithm does not require any future knowledge of RE sources. They have discussed three benchmark algorithms: FABEF, RR, and HAREF. FABEF matches a UR to all the available datacenters and assigns it to the least cost datacenter. RR assigns the URs to the datacenters in a circular fashion. HAREF matches a UR to all the available datacenters and assigns it to the highest available RE resources datacenter. They have observed that HAREF utilizes maximum RE resources in compared to FABEF and FLB. Nayak et al. [31] have taken the availability of resources and energy cost into account and presented a multi-objective algorithm by considering their linear combination. They have performed normalization of cost and resource availability to avoid dominance. The above-discussed algorithms do not consider the UN and assume that the URs finish their execution without any interruption. The comparison of these algorithms is shown in Table 1.

This paper has the following likeness and unlikeness to the recent literature.

1. Unlike [17, 20, 21, 24, 31], the proposed algorithms consider UN of RE and NRE. UN of RE is made variable to make it more practicable.

Table 1 Comparison of existing and proposed algorithms

Algorithm	UN	Focus on Cost	Focus on RE	Matching Decision	Ease of Implementation
FABEF [17]	×	✓	×	✓	×
SCA [20]	×	✓	×	✓	×
RR [17]	×	×	×	×	✓
HAREF [17]	×	×	✓	✓	×
MB [21]	×	×	✓	✓	×
WF [20]	×	×	×	✓	×
UN-FABEF	✓	✓	×	✓	×
UN-RR	✓	×	×	×	✓
UN-HAREF	✓	×	✓	✓	×

2. Unlike [17, 20, 21, 24, 31], the URs do not complete their execution as per the mentioned duration. This is due to managing the UN.
3. Like [17, 31], the proposed algorithms take their matching decision.
4. Like [17, 31], the performance of the proposed algorithms is assessed in terms of overall cost and the number of used RE resources. Additionally, the performance is assessed using UN cost and time.

3 Renewable energy-based system model and problem formulation

This section discusses the system model and presents the problem with objectives and constraints.

3.1 System model

Consider a CSP that hosts a set of datacenters to assign resources to the URs as per their SLA requirements. These datacenters are powered by both RE and NRE sources. However, the CSP tries to balance the power requirement of the resources of the datacenters by RE sources as it is sustainable and environment-friendly. In case of further power requirements, it can be provided using NRE sources. In the view of CSP, the cost of using RE resources is negligible, whereas the cost of using NRE resources varies with respect to time as it relies on electricity generation costs. On the contrary, in the view of the user, the cost of using RE resources is cheaper than NRE resources. However, the UN of using RE resources is more compared to the NRE resources. In the proposed model, the UR is matched with all the available datacenters to determine the number of RE and NRE resource slots as per its requirements. Subsequently, UN can be determined using these resource slots. Finally, the assignment cost of the UR is determined in each datacenter, including UN time duration. In the case of UN-FABEF, the UR is assigned to the datacenter with the least cost. On the contrary, the UN-HAREF assigns the UR to the datacenter with the highest RE resource slots. However, UN-RR assigns the UR to the datacenter without considering the assignment cost. The above model is an extension of the model considered in [17, 31].

3.2 Problem formulation

Consider a set of n URs, $UR = \{UR_1, UR_2, UR_3, \ldots, UR_n\}$ and a set of m datacenters, $DC = \{DC_1, DC_2, DC_3, \ldots, DC_m\}$. Note that $n \gg m$. Each UR, $UR_k, 1 \leq k \leq n$, is represented using a 6-tuple, i.e., $UR_k = <UR\#, ST, D, N, UN\text{-}RE, UN\text{-}NRE>$. Here, UR # represents the unique identification number of UR_k, ST represents start time of UR_k, D represents duration of UR_k, N represents the number of nodes/resources required for UR_k, *UN-RE* represents the *UN* of RE resources available for UR_k and *UN-NRE* represents the *UN* of NRE resources available for UR_k. *UN-RE* and *UN-NRE* can be any value between 1% and 100%. Mathematically, *UN-RE* = [1%–100%] and *UN-NRE* = [1%–100%]. In the best-case scenario, the value of *UN-RE* and *UN-NRE* is 0%. However, in the worst-case scenario, the value of *UN-RE* and *UN-NRE* is 100%. Each datacenter, $DC_i, 1 \leq i \leq m$, is represented by using a 5-tuple, i.e., $DC_i = <DC\#, R, ARE, ANRE, CO>$. Here, DC # represents the unique identification number of DC_i, R represents a number of resources in DC_i, ARE represents a series of available RE resources in DC_i over a period of time, $ANRE$ represents a series of available NRE resources in DC_i over a period of time and CO represents the cost of RE and NRE resources of DC_i, respectively. The sum of RE (i.e., re) and NRE (i.e., nre) resources is R. Mathematically, $R = re + nre$.

The problem is to match each UR, one by one, with all the available datacenters and schedule the UR to a suitable datacenter, such that the following objectives are fulfilled.

1. The overall cost is minimized.
2. The *UN* cost and time are minimized.
3. The number of used RE resources is maximized.

This problem is formulated with the following constraints.

1. The order of URs remains intact.
2. The number of nodes can be provided from either RE or NRE resources or both, and it cannot exceed the maximum available resources at any particular time.
3. The duration of UR can be adjusted by incorporating the *UN*.

4 Proposed scheduling algorithms

This section presents three proposed algorithms: UN-FABEF, UN-RR, and UN-HAREF. These algorithms aim to minimize the overall cost, UN cost and UN time, and maximize the number of used RE resources. The pseudo-codes for UN-FABEF, UN-RR and UN-HAREF are shown in Algorithms 1 to 3, respectively. The list of symbols and their definitions are shown in Table 2.

4.1 UN-FABEF

UN-FABEF picks a UR (say, UR_k) from the queue of URs (*URQ*) (Line 1 and Line 3 of Algorithm 1) and matches the UR with all the available datacenters (Line 4). In the process of matching, it finds the resource slots that are available between *ST* of UR_k, and the sum of *ST* and *D* of UR_k minus one, as per the requirement of UR_k (Line 6). Then it determines the number of RE (i.e., *slots_re*) and NRE (i.e., *slots_nre*) resource slots that can be given to that UR and the assignment cost without *UN* (Line 8 to Line 21). If the number of resource slots is equal to the number of resource slots required for that UR at a time instance (Line 18), then it breaks the loop (Line 19) and continues with another time instance (Line 6). Otherwise, it will be an infinite loop. Next, UN-FABEF calls the Procedure 1 (*DETERMINE-UNCERTAINTY*) to determine the *UN* (Line 23). *UN* can be determined as follows (Line 1 of Procedure 1).

$$UN[k] = \lceil (re \times UN\text{-}RE[k] + nre \times UN\text{-}NRE[k]) \rceil, 1 \le k \le n \tag{1}$$

where $UN\text{-}RE[k]$ and $UN\text{-}NRE[k]$ are *UN* of RE and NRE for UR_k. The range of *UN-RE* and *UN-NRE* is between 1% and 100%. As the number of resource slots is an integer, we use ceil function ($\lceil\ \rceil$) to round the value to the next integer.

Next, the duration of UR_k is extended from $ST[k] + D[k]$ - 1 to $ST[k] + D[k] + \lceil \frac{UN[k]}{N[k]} \rceil$ by incorporating the *UN* (Line 2). Then it adds the number of RE and NRE resource slots that can be required to fulfil the *UN* and calculates the updated assignment cost by considering *UN* (Line 4 to Line 17). This procedure is called for each UR to determine the updated assignment cost.

Now, the least assignment cost is calculated and the corresponding datacenter is identified (Line 25 of Algorithm 1). Finally, the UR is assigned to the least assignment cost datacenter and the overall cost is updated (Line 26 and Line 27).

Table 2 Symbols and their definition

Symbols	Definitions
n	Number of URs
m	Number of datacenters
o	Number of resources
UR_k	*k*th user request
DC_i	*i*th datacenter
ST	Start time
D	Duration
N	Number of nodes/resources
UN-RE	*UN* of RE resources
UN-NRE	*UN* of NRE resources
ARE	Available RE resources
ANRE	Available NRE resources
CO	Cost of RE and NRE resources
OCO	Overall cost of RE and NRE resources
ACO	Assignment cost of RE and NRE resources
UNCO	*UN* cost of RE and NRE resources
UNT	*UN* time
URE	Number of used RE resources
R	Number of resources
re	Number of RE resources
nre	Number of NRE resources
slots_re	Number of RE resource slots
slots_nre	Number of NRE resource slots
URQ	Queue of URs

Algorithm 1 Pseudo-code for UN-FABEF

Input: URQ, ST, D, N, R, CO, n, m, o
Output: OCO and number of used RE resources

```
1: while $URQ \neq NULL$ do                                  ▷ $URQ$ = Queue of URs
2:     Set $OCO = 0$                                          ▷ $OCO$ = Overall cost
3:     for $k$ = 1, 2, 3,..., $n$ do                          ▷ $n = |URQ|$
4:         for $i$ = 1, 2, 3,..., $m$ do
5:             Set $ACO[i] = 0$, $slots\_re = 0$ and $slots\_nre = 0$ ▷ $ACO$ = Assignment cost
6:             for $l = ST[k], ST[k] + 1, ST[k] + 2, \ldots, ST[k] + D[k] - 1$ do
7:                 Set $slots = 0$
8:                 for $j$ = 1, 2, 3,..., $o$ do
9:                     if $R[l, j]$ is not assigned to any UR then
10:                        $slots$ += 1
11:                        if $R[l, j]$ is powered by the RE sources then
12:                            $slots\_re$ += 1             ▷ $slots\_re$ = Number of RE slots
13:                        else
14:                            $slots\_nre$ += 1            ▷ $slots\_nre$ = Number of NRE slots
15:                            $ACO[i]$ += $CO[l]$
16:                        end if
17:                    end if
18:                    if $slots == N[k]$ then
19:                        break
20:                    end if
21:                end for
22:            end for
23:            Call $DETERMINE\text{-}UNCERTAINTY(k, UN\text{-}RE, UN\text{-}NRE, ST, D, N, R, o, re, nre, ACO, CO)$
24:        end for
25:        Find $min(ACO)$ and determine the best datacenter $i'$ that holds the minimum value   ▷ $min$ is a function to find the minimum cost
26:        Assign the UR $k$ to the datacenter $i'$
27:        $OCO$ += $ACO[i']$ and update the number of used RE resources
28:    end for
29: end while
```

4.2 UN-RR

UN-RR picks a UR (say, UR_k) from the URQ (Line 1 and Line 3 of Algorithm 2). Then it determines the datacenter by finding whether k is a multiple of m (Line 4). In the process of matching, it finds the resource slots that are available between ST of UR_k, and the sum of ST and D of UR_k minus one (Line 6). Then it determines the *slots_re* and *slots_nre* that can be given to that UR and the overall cost (Line 8 to Line 21). We use the break statement in line 19 to exit from the line 8 inner for loop. This is done when a datacenter in a particular time slot fulfils the node requirements of a UR. Next, UN-RR calls the Procedure 1 (*DETERMINE-UNCERTAINTY*) to determine the *UN* (Line 23). Note the procedure is the same as the UN-FABEF. Hence, it is not explicitly shown here. However, the symbol *ACO* needs to be replaced with *OCO* in line 11 of Procedure 1.

Procedure 1 *DETERMINE-UNCERTAINTY* (k, $UN\text{-}RE$, $UN\text{-}NRE$, ST, D, N, R, o, re, nre, ACO, CO)

```
1: $UN[k] = \lceil re \times UN\text{-}RE[k] + nre \times UN\text{-}NRE[k] \rceil$
2: for $l = ST[k] + D[k]$, $ST[k] + D[k] + 1, \ldots, ST[k] + D[k] + \lceil \frac{UN[k]}{N[k]} \rceil$ do
3:     Set $slots = 0$
4:     for $j = 1, 2, 3, \ldots, o$ do
5:         if $R[l, j]$ is not assigned to any UR then
6:             $slots$ += 1
7:             if $R[l, j]$ is powered by the RE sources then
8:                 $slots\_re$ += 1
9:             else
10:                $slots\_nre$ += 1
11:                $ACO[i]$ += $CO[l]$
12:            end if
13:        end if
14:        if $slots == N[k]$ then
15:            break
16:        end if
17:    end for
18: end for
```

Algorithm 2 Pseudo-code for UN-RR

Input: URQ, ST, D, N, R, CO, n, m, o
Output: OCO and number of used RE resources

```
1: while $URQ \neq NULL$ do
2:     Set $OCO = 0$
3:     for $k = 1, 2, 3, \ldots, n$ do
4:         ($k$ is a multiple of $m$) ? $i = m : i = k \bmod m$
5:         $ACO[i] = 0$, $re = 0$ and $nre = 0$
6:         for $l = ST[k]$, $ST[k] + 1$, $ST[k] + 2, \ldots, ST[k] + D[k]$ - 1 do
7:             Set $slots = 0$
8:             for $j = 1, 2, 3, \ldots, o$ do
9:                 if $R[l, j]$ is not assigned to any UR then
10:                    $slots$ += 1
11:                    if $R[l, j]$ is powered by the RE sources then
12:                        $slots\_re$ += 1
13:                    else
14:                        $slots\_nre$ += 1
15:                        $OCO[i]$ += $CO[l]$
16:                    end if
17:                end if
18:                if $slots == N[k]$ then
19:                    break
20:                end if
21:            end for
22:        end for
23:        Call $DETERMINE\text{-}UNCERTAINTY(k, UN\text{-}RE, UN\text{-}NRE, ST, D, N, R,$
           $o, re, nre, ACO, CO)$
24:    end for
25:    Update the number of used RE resources
26: end while
```

4.3 UN-HAREF

UN-HAREF picks a UR (say, UR_k) from the URQ (Line 1 and Line 3 of Algorithm 3) and matches the UR with all the available datacenters (Line 4). In the process of matching, it finds the resource slots that are available between ST of UR_k, and the sum of ST and D of UR_k minus one (Line 6). Then it determines the *slots_re* and *slots_nre* that can be given to that UR and the assignment cost without UN (Line 8 to Line 21). We exit from the inner for loop (line 8) using the break statement in line 19. This is performed when the node requirements of a UR are fulfilled by a datacenter in a particular time slot. Next, UN-HAREF calls the Procedure 2 (*DETERMINE-UNCERTAINTY-HAREF*) to determine the *UN* (Line 23). Like UN-FABEF, the duration of UR_k is extended from $ST[k] + D[k]$ - 1 to $ST[k] + D[k] + \lceil \frac{UN[k]}{N[k]} \rceil$ by incorporating the *UN* (Line 2). Then it adds the *slots_re* and *slots_nre* that can be required to fulfil the *UN* and calculates the updated assignment cost by considering *UN* (Line 4 to Line 17).

Now the datacenter with the highest *slots_re* is identified (Line 25 of Algorithm 3). Finally, the UR is assigned to the datacenter with the highest *slots_re*, and the overall cost is updated (Line 26 and Line 27).

4.4 Time complexity

Let *n* be the number of UR, *m* be the number of datacenters, *o* be the number of resources, and *d* be the maximum duration of all the URs. In the UN-FABEF, the process of matching for each request takes *O*(*dmo*) time. Procedure 1

Algorithm 3 Pseudo-code for UN-HAREF

```
Input: $URQ$, $ST$, $D$, $N$, $R$, $CO$, $n$, $m$, $o$
Output: $OCO$ and number of used RE resources
 1: while $URQ \neq NULL$ do
 2:     Set $OCO = 0$
 3:     for $k = 1, 2, 3, \dots, n$ do
 4:         for $i = 1, 2, 3, \dots, m$ do
 5:             Set $ACO[i] = 0$, $slots\_re[i] = 0$ and $slots\_nre[i] = 0$
 6:             for $l = ST[k], ST[k] + 1, ST[k] + 2, \dots, ST[k] + D[k]$ - 1 do
 7:                 Set $slots = 0$
 8:                 for $j = 1, 2, 3, \dots, o$ do
 9:                     if $R[l, j]$ is not assigned to any UR then
10:                         $slots$ += 1
11:                         if $R[l, j]$ is powered by the RE sources then
12:                             $slots\_re[i]$ += 1
13:                         else
14:                             $slots\_nre[i]$ += 1
15:                             $ACO[i]$ += $CO[l]$
16:                         end if
17:                     end if
18:                     if $slots == N[k]$ then
19:                         break
20:                     end if
21:                 end for
22:             end for
23:             Call $DETERMINE\text{-}UNCERTAINTY\text{-}HAREF(k, UN\text{-}RE, UN\text{-}NRE,$
    $ST, D, N, R, o, re, nre, ACO, CO)$
24:         end for
25:         Find $max(slots\_re)$ and determine the best DC $i'$ that holds the maximum value
    ▷ $max$ is a function to find the maximum cost
26:         Assign the UR $k$ to the DC $i'$
27:         $OCO$ += $ACO[i']$
28:     end for
29: end while
```

Procedure 2 *DETERMINE-UNCERTAINTY-HAREF*(k, $UN\text{-}RE$, $UN\text{-}NRE$, ST, D, N, R, o, re, nre, ACO, CO)

```
1: $UN[k] = \lceil re[i] \times UN\text{-}RE[k] + nre[i] \times UN\text{-}NRE[k] \rceil$
2: for $l = ST[k] + D[k]$, $ST[k] + D[k] + 1, \ldots, ST[k] + D[k] + \lceil \frac{UN[k]}{N[k]} \rceil$ do
3:     Set $slots = 0$
4:     for $j = 1, 2, 3, \ldots, o$ do
5:         if $R[l, j]$ is not assigned to any UR then
6:             $slots$ += 1
7:             if $R[l, j]$ is powered by the RE sources then
8:                 $slots\_re[i]$ += 1
9:             else
10:                $slots\_nre[i]$ += 1
11:                $ACO[i]$ += $CO[l]$
12:            end if
13:        end if
14:        if $slots == N[k]$ then
15:            break
16:        end if
17:    end for
18: end for
```

takes $O(do)$ time for each UR. The process of scheduling for each request takes $O(m)$ time. Therefore, the overall time complexity of UN-FABEF is $O(dmno)$ time for executing all the URs. In the UN-RR, the process of matching for each request takes $O(do)$ time. Procedure 1 takes $O(do)$ time for each UR. The process of scheduling for each request takes $O(1)$ time. Therefore, the overall time complexity of UN-RR is $O(dno)$ time for executing all the URs. Note that the selection of the datacenter in UN-RR takes $O(1)$ time. Like UN-FABEF, the overall time complexity of UN-HAREF is $O(dmno)$ time for executing all the URs.

4.5 Illustration

Let us consider that there are nine URs (i.e., UR_1 to UR_9) with their tuple (i.e., *ST*, *D*, *N*, *UN-RE* and *UN-NRE*) as shown in Table 3 and these URs need to be matched to two datacenters (i.e., DC_1 and DC_2) in order to assign to one of the datacenters. Note that we use different colours for URs to uniquely identify each pictorially during the matching process and in the Gantt chart. An initial setup of two datacenters and their resource is shown in Table 4. Here, we show seven resources in each datacenter for the simplicity of illustration, and it is shown in the second column of the same table. The first row and last row of each datacenter show the cost of the NRE resources and time slots. For instance, the cost of the NRE resources in DC_1 is 0.3 at time slot t = 3, whereas the cost of the NRE resources in DC_2 is 0.5 at the same time slot. The green colour in each datacenter represents the RE resources, and the white colour represents the NRE resources, respectively. As seen in this table, the RE resources of each datacenter fluctuate over the time slots. The cost of RE resources is cheaper in comparison to NRE resources. Therefore, it is assumed to be negligible for easy illustration.

4.6 Illustration of UN-FABEF

In the UN-FABEF, the first UR (i.e., UR_1) requires one node for the time slots t = 1 to t = 4 (i.e., duration of 4 time slots) without *UN*. The *UN* of RE for this UR is pre-determined as 50%, and NRE is pre-determined as 10%. The UR_1 is first matched with DC_1 as shown in Table 5 in which we underline (_) the RE resource slots occupied by that UR. Here, DC_1 can provide four RE resource slots. Therefore, the finish time of UR_1 in DC_1 is determined as $4 + \lceil(4 \times 50\%)\rceil = 6$ with *UN*, and the assignment cost is calculated as negligible (i.e., 0). Then UR_1 is matched DC_2 as shown in Table 5. Like DC_1, DC_2 can provide four RE resource slots. Therefore, the finish time of UR_1 in DC_2 is determined as $4 + \lceil(4 \times 50\%)\rceil = 6$. However, the assignment cost is calculated as 0.3. This cost is due to the NRE resource slot at t = 6. As DC_1 takes least assignment cost, UR_1 is assigned to it.

Next, UR_2 requires one node at the time slot t = 1 without *UN*. The *UN* of RE for this UR is pre-determined as 100%, and NRE is pre-determined as 10%. The UR_2 is first matched with DC_1 as shown in Table 6 in which we underline (_) the RE resource slots occupied by that UR. Here, DC_1 can provide one RE resource slot. Therefore, the finish time of UR_2 in DC_1 is determined as $1 + \lceil(1 \times 100\%)\rceil = 2$ with *UN*, and the assignment cost is calculated as 0.1. This cost is due to the NRE resource slot at t = 2. Then UR_2 is matched DC_2 as shown in Table 6. Like DC_1, DC_2 can provide one RE resource slot. Therefore, the finish time of UR_2 in DC_2 is determined as $1 + \lceil(1 \times$

Table 3 A set of nine URs with their tuple

UR #	ST	D	N	UN-RE	UN-NRE
UR_1	1	4	1	50%	10%
UR_2	1	1	1	100%	10%
UR_3	1	4	1	25%	10%
UR_4	3	5	2	60%	10%
UR_5	4	3	1	33%	10%
UR_6	5	2	1	50%	10%
UR_7	5	3	1	33%	10%
UR_8	7	2	2	100%	10%
UR_9	8	2	3	50%	10%

$100\%)\rceil = 2$. However, the assignment cost is calculated as 0. As DC_2 takes least assignment cost, UR_2 is assigned to it.

Next, UR_3 requires one node for the time slots t = 1 to t = 4 without UN. The UN of RE for this UR is pre-determined as 25%, and NRE is pre-determined as 10%. The UR_3 is first matched with DC_1 as shown in Table 7. Here, DC_1 can provide three RE resource slots and one NRE slot. Therefore, the finish time of UR_3 in DC_1 is determined as $4 + \lceil (3 \times 25\%) + (1 \times 10\%) \rceil = 5$ with UN and the assignment cost is calculated as 0.1. This cost is due to the NRE resource slot at t = 2. Then UR_3 is matched DC_2 as shown in Table 7. Like DC_1, DC_2 can provide two RE resource slots and two NRE resource slots. Therefore, the finish time of UR_3 in DC_2 is determined as $4 + \lceil (2 \times 25\%) + (2 \times 10\%) \rceil = 5$. However, the assignment cost is calculated as 0.2. This cost is due to the NRE resource slots at t = 1 and t = 2. As DC_1 takes least assignment cost, UR_3 is assigned to it.

Next, UR_4 requires 2 nodes for the time slots t = 3 to t = 7 without UN. The UN of RE for this UR is pre-determined as 60%, and NRE is pre-determined as 10%. The UR_4 is first matched with DC_1. Here, DC_1 can provide seven RE resource slots and three NRE resource slots. The UN time is $\lceil (7 \times 60\%) + (3 \times 10\%) \rceil$ = 5 for 2 nodes. Therefore, the finish time of UR_4 in DC_1 is determined as $5 + \lceil \frac{5}{2} \rceil = 8$ with UN and the assignment cost is calculated as 2.3. Then UR_4 is matched DC_2. Here, DC_2 can provide eight RE resource slots and two NRE resource slots. The UN time is $\lceil (8 \times 60\%) + (2 \times 10\%) \rceil = 5$ for 2 nodes. Therefore, the finish time of UR_4 in DC_2 is determined as $5 + \lceil \frac{5}{2} \rceil = 8$. However, the assignment cost is calculated as 0.8. As DC_2 takes least assignment cost, UR_4 is assigned to it. In the similar process, UR_5 to UR_9 are assigned to DC_1, DC_1, DC_1, DC_1 and DC_1, respectively. The assignment costs are 0.6, 0.6, 0.6, 0.6 and 0.9, respectively. The overall cost of DC_1 and DC_2 is 3.4 and 0.8, respectively. The number of used RE resources is 33. Note that the UN cost and time are 0.5 and 12, respectively. The final Gantt chart for UN-FABEF is shown in Table 8.

4.7 Illustration of UN-RR

In the UN-RR, the first UR (i.e., UR_1) is assigned to DC_1 as 1 % 2 = 1. Here, DC_1 can provide four RE resource slots. However, the finish time of UR_1 in DC_1 is determined as $4 + \lceil (4 \times 50\%) \rceil = 6$ with UN, and the assignment cost is calculated as 0. Next, UR_2 is assigned to DC_2. Here, DC_2 can provide one RE resource slot. However, the finish time of UR_2 in DC_2 is determined as $1 + \lceil (1 \times 100\%) \rceil = 2$ with UN, and the assignment cost is calculated as 0. Then UR_3 is assigned to DC_1 as 3 % 2 = 1. Here, DC_1 can

Table 4 An initial setup of datacenters and their resource, and cost of the NRE resources

	R #	0.1	0.1	0.3	0.4	0.4	0.2	0.2	0.1	0.1	0.3
DC_1	7										
	6										
	5										
	4										
	3				🟩			🟩			
	2	🟩		🟩	🟩	🟩		🟩		🟩	🟩
	1	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
		$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$	$t = 7$	$t = 8$	$t = 9$	$t = 10$
	R #	0.1	0.1	0.5	0.3	0.2	0.3	0.4	0.5	0.1	0.3
DC_2	7										
	6										
	5										
	4										
	3										
	2			🟩	🟩	🟩		🟩	🟩		🟩
	1	🟩	🟩	🟩	🟩	🟩		🟩	🟩		🟩
		$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$	$t = 7$	$t = 8$	$t = 9$	$t = 10$

provide three RE resource slots and one NRE resource slot. However, the finish time of UR_3 in DC_1 is determined as $4 + \lceil(3 \times 25\%) + (1 \times 10\%)\rceil = 5$ with UN and the assignment cost is calculated as 0.1. This cost is due to the NRE resource slot at t = 2.

Next, UR_4 is assigned to DC_2. Here, DC_2 can provide two NRE resource slots and eight RE resource slots. The UN time is $\lceil(8 \times 60\%) + (2 \times 10\%)\rceil = 5$ for 2 nodes. Therefore, the finish time of UR_4 in DC_2 is determined as $5 + \lceil\frac{5}{2}\rceil = 8$ with UN and the assignment cost is calculated as 0.8. In the similar process, UR_5 to UR_9 are assigned to DC_1, DC_2, DC_1, DC_2 and DC_1, respectively. The assignment costs are 0.6, 0.9, 0.6, 2.0 and 0.7, respectively. The overall cost of DC_1 and DC_2 is 2.0 and 3.7, respectively. The number of used RE resources is 32. Note that the UN cost and time are 0.5 and 12, respectively. The final Gantt chart for UN-RR is shown in Table 9.

4.8 Illustration of UN-HAREF

In the UN-HAREF, the first UR (i.e., UR_1) requires one node for the time slots t = 1 to t = 4 (i.e., duration of 4 time slots) without UN. The UN of RE for this UR is pre-determined as 50%, and NRE is pre-determined as 10%. The UR_1 is first matched with DC_1. Here, DC_1 can provide four RE resource slots. Therefore, the finish time of UR_1 in DC_1 is determined as 4 + (4 × 50%) = 6 with UN in which all the resource slots are RE. Here, the assignment cost is calculated as 0. Then UR_1 is matched DC_2. Like DC_1, DC_2 can provide four RE resource slots. Therefore, the finish time of UR_1 in DC_2 is determined as 4 + (4 × 50%) = 6 in which five resource slots are RE and one resource slot is NRE. Here, the assignment cost is calculated as 0.3. As DC_1 gives highest RE resource slots, UR_1 is assigned to it.

Next, UR_2 requires one node at the time slot t = 1 without UN. The UN of RE for this UR is pre-determined as 100%, and NRE is pre-determined as 10%. The UR_2 is first matched with DC_1. Here, DC_1 can provide one RE resource slot. Therefore, the finish time of UR_2 in DC_1 is determined as 1 + (1 × 100%) = 2 with UN in which one resource slot is RE, and another resource slot is NRE. Here, the assignment cost is calculated as 0.1. Then UR_2 is matched DC_2. Like DC_1, DC_2 can provide one RE resource slot. Therefore, the finish time of UR_2 in DC_2 is determined as 1 + (1 × 100%) = 2 in which two resource slots are RE. Here, the assignment cost is calculated as 0. As DC_2 gives highest RE resource slots, UR_2 is assigned to it.

Next, UR_3 requires one node for the time slots t = 1 to t = 4 without UN. The UN of RE for this UR is pre-determined as 25%, and NRE is pre-determined as 10%. The UR_3 is first matched with DC_1. Here, DC_1 can provide three RE resource slots and one NRE slot. Therefore, the finish time of UR_3 in DC_1 is determined as 4 + (3 × 25%) + (1 × 10%) = 5 with UN in which four resource slots are RE, and one resource slot is NRE. Here, the assignment cost is calculated as 0.1. Then UR_3 is matched DC_2. Like DC_1, DC_2 can provide two RE resource slots and two NRE resource slots. Therefore, the finish time of UR_3 in DC_2 is determined as 4 + (2 × 25%) + (2 × 10%) = 5 in which three resource slots are RE and two resource slots are NRE. Here, the assignment cost is calculated as 0.2. As DC_1 gives highest RE resource slots, UR_3 is assigned to it.

Next, UR_4 requires 2 nodes for the time slots t = 3 to t = 7 without UN. The UN of RE for this UR is pre-determined as 60%, and NRE is pre-determined as 10%. The UR_4 is first matched with DC_1. Here, DC_1 can provide seven RE resource slots and three NRE resource slots. The UN time is $\lceil(7 \times 60\%) + (3 \times 10\%)\rceil$ = 5 for 2 nodes. Therefore, the finish time of UR_4 in DC_1 is determined as $5 + \lceil\frac{5}{2}\rceil$ = 8 with UN in which twelve resource slots are RE, and four resource slots are NRE. Here, the assignment cost is calculated as 2.3. Then UR_4 is matched DC_2. Here, DC_2 can provide eight RE resource slots and two NRE resource slots. The UN time is $\lceil(8 \times 60\%) + (2 \times 10\%)\rceil$ = 5 for 2 nodes. Therefore, the finish time of UR_4 in DC_2 is determined as $5 + \lceil\frac{5}{2}\rceil = 8$ in which twelve resource slots are RE and four resource slots are NRE. Here, the assignment cost is calculated as 0.8. Both DC_1 and DC_2 give the same number of RE resource slots, UR_4 is assigned to DC_2 based on least assignment cost. In the similar process, UR_5 to UR_9 are assigned to DC_1, DC_1, DC_1, DC_1 and DC_1, respectively. The assignment costs are 0.6, 0.6, 0.6, 0.6 and 0.9, respectively. The overall assignment cost of DC_1 and DC_2 is 3.4 and 0.8, respectively. The number of used RE resources is 33. Note that the UN cost and time are 0.5 and 12, respectively. The final Gantt chart for UN-HAREF is shown in Table 10.

Table 5 Matching of UR_1 to DC_1 and DC_2 to determine its assignment cost

	R #	0.1	0.1	0.3	0.4	0.4	0.2	0.2	0.1	0.1	0.3
DC_1	7										
	6										
	5										
	4										
	3				🟩			🟩			
	2	🟩		🟩	🟩	🟩		🟩		🟩	🟩
	1	UR_1	UR_1	UR_1	UR_1	UR_1	UR_1	🟩	🟩	🟩	🟩
		$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$
	R #	0.1	0.1	0.5	0.3	0.2	0.3	0.4	0.5	0.1	0.3
DC_2	7										
	6										
	5										
	4										
	3										
	2			🟩	🟩	🟩		🟩	🟩		🟩
	1	UR_1	UR_1	UR_1	UR_1	UR_1	UR_1	🟩	🟩		🟩
		$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$

5 Performance metrics, simulation configuration and simulation results

This section discusses the performance metrics used to evaluate the proposed algorithms, the simulation configuration and the simulation results.

5.1 Performance metrics

We use four performance metrics, namely overall cost, UN cost and time, and the number of used RE resources. The overall cost (OCO) is the sum of the cost incurred for using RE and NRE resources in all the datacenters. However, the cost of RE resources is assumed to be negligible. Mathematically,

$$OCO = \sum_{k=1}^{n}\sum_{i=1}^{m} ACO[i] \times X[k,i] \tag{2}$$

$$\text{where } X[k,i] = \left\{ \begin{array}{ll} 1 & \text{if } UR_k \text{ is assigned to } DC_i \\ 0 & \text{Otherwise} \end{array} \right\}$$

Note that resources are not explicitly shown here. The UN cost ($UNCO$) is the sum of the cost incurred for using RE and NRE resources in the UN time duration. Note that the time duration is between $ST + D$ and $ST + D + \lceil \frac{UN}{N} \rceil$. Mathematically,

$$UNCO = \sum_{k=1}^{n}\sum_{i=1}^{m} \sum_{l=ST[k]+D[k]}^{ST[k]+D[k]+\lceil \frac{UN[k]}{N[k]} \rceil} CO[l] \times X[k,i] \tag{3}$$

The UN time (UNT) is the sum of the extended time duration of URs to fulfil their UN. Mathematically,

$$UNT = \sum_{k=1}^{n} UN[k] \tag{4}$$

The number of used RE resources (URE) is the sum of RE resource slots that are assigned to the URs.

$$URE = \sum_{k=1}^{n} slots\_re[k] \tag{5}$$

5.2 Simulation configuration

The simulation is carried out in a system with the following configuration. 1) System processor: Intel(R) Core(TM) i3-7020U CPU @ 2.30 GHz 2.30 GHz 2) Installed system RAM: 4.00 GB 3) System type: x64-based processor 4) System operating system: 64-bit 5) Windows edition: Windows 10 Home 6) System software: MATLAB R2021a. As there are no benchmark datasets, we generate datasets by taking the number of URs in the range of [200–2000] and the number of datacenters in the range of [20–200]. Alternatively, we generate ten datasets, namely 200×20, 400×40, 600×60, 800×80, 1000×100, 1200×120, 1400×140, 1600×160, 1800×180 and 2000×200 using uniformly distributed pseudorandom integers. Note the first values (i.e., 200, 400, 600,. . ., 2000) represent the number of URs, and the second values represent the number of datacenters. In each dataset, there are five instances, namely $i1$ to $i5$. These instances are generated in the same range. The results of these instances are averaged and shown as the result of the dataset. We consider 10 to 400 resources per datacenter. It indicates that the number of resources in the dataset ranges from 200 to 80,000. The cost of using NRE resources is generated in

Table 6 Matching of UR_2 to DC_1 and DC_2 to determine its assignment cost

	R #	0.1	0.1	0.3	0.4	0.4	0.2	0.2	0.1	0.1	0.3
DC_1	7										
	6										
	5										
	4										
	3				🟩			🟩			
	2	$\underline{UR_2}$	$\underline{UR_2}$	🟩	🟩	🟩		🟩		🟩	🟩
	1	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	🟩	🟩	🟩	🟩
		$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$
DC_2	R #	0.1	0.1	0.5	0.3	0.2	0.3	0.4	0.5	0.1	0.3
	7										
	6										
	5										
	4										
	3										
	2			🟩	🟩	🟩		🟩	🟩		🟩
	1	$\underline{UR_2}$	$\underline{UR_2}$	🟩	🟩	🟩		🟩	🟩		🟩
		$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$

Table 7 Matching of UR_3 to DC_1 and DC_2 to determine its assignment cost

	R #	0.1	0.1	0.3	0.4	0.4	0.2	0.2	0.1	0.1	0.3
DC_1	7										
	6										
	5										
	4										
	3				🟩			🟩			
	2	$\underline{UR_3}$	$\underline{UR_3}$	$\underline{UR_3}$	$\underline{UR_3}$	$\underline{UR_3}$		🟩		🟩	🟩
	1	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	🟩	🟩	🟩	🟩
		$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$
DC_2	R #	0.1	0.1	0.5	0.3	0.2	0.3	0.4	0.5	0.1	0.3
	7										
	6										
	5										
	4										
	3										
	2	$\underline{UR_3}$	$\underline{UR_3}$	🟩	🟩	🟩		🟩	🟩		🟩
	1	$\underline{UR_2}$	$\underline{UR_2}$	$\underline{UR_3}$	$\underline{UR_3}$	$\underline{UR_3}$		🟩	🟩		🟩
		$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$

the range of [1–100], and the cost of RE resources is assumed as 0. The parameters of URs are configured as follows. (1) *ST*: [1–100], (2) *D*: [10–25], (3) *N*: [10–100], (4) *UN-RE*: [10–90], (5) *UN-NRE*: 10%. It is noteworthy to mention that all these parameters are combined together and given in a single file for all the URs. In a similar fashion, all the parameters associated with the datacenters are given in another file. Each algorithm takes these files to produce the simulation results using four performance metrics.

5.3 Simulation results

In the simulation process, the results of the proposed algorithms are carried out using the generated datasets. The overall cost of the UN-FABEF, UN-RR, and UN-HAREF algorithms is compared and shown in Fig. 1. Here, the *x*-axis represents the datasets and the *y*-axis represents the overall cost on the logarithmic scale. As seen in the figure, UN-FABEF achieves the least overall cost compared to UN-RR and UN-HAREF. The rationality behind this performance is that it assigns the URs to the least assignment cost datacenter. It is also important to notice that UN-RR achieves better performance than UN-HAREF, even if the scheduling is performed in a circular fashion. On the contrary, UN-HAREF performs poorly once the RE resource slots are occupied. Therefore, the overall cost of UN-HAREF is drastically increased.

The *UN* cost and time of the UN-FABEF, UN-RR, and UN-HAREF algorithms are compared and shown in Figs. 2

Table 8 Final Gantt chart of assigning the URs to the DCs using UN-FABEF

	R #	0.1	0.1	0.3	0.4	0.4	0.2	0.2	0.1	0.1	0.3
	7										
	6								UR_9		
	5					UR_7		UR_8	UR_9	UR_9	
DC_1	4					UR_6	UR_7	UR_8	UR_9	UR_9	
	3				UR_5	UR_5	UR_6	UR_7	UR_8	UR_9	UR_9
	2	UR_3	UR_3	UR_3	UR_3	UR_3	UR_5	UR_6	UR_8	UR_8	UR_9
	1	UR_1	UR_1	UR_1	UR_1	UR_1	UR_1	UR_5	UR_7	UR_8	UR_9
		$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$
	R #	0.1	0.1	0.5	0.3	0.2	0.3	0.4	0.5	0.1	0.3
	7										
	6										
	5										
DC_2	4										
	3										
	2			UR_4	UR_4	UR_4	UR_4	UR_4	UR_4	UR_4	UR_4
	1	UR_2	UR_2	UR_4	UR_4	UR_4	UR_4	UR_4	UR_4	UR_4	UR_4
		$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$

Table 9 Final Gantt chart of assigning the URs to the DCs using UN-RR

	R #	0.1	0.1	0.3	0.4	0.4	0.2	0.2	0.1	0.1	0.3
	7										
	6										
	5										
DC_1	4					UR_7			UR_9		
	3				UR_5	UR_5	UR_7		UR_9	UR_9	UR_9
	2	UR_3	UR_3	UR_3	UR_3	UR_3	UR_5	UR_7	UR_9	UR_9	UR_9
	1	UR_1	UR_1	UR_1	UR_1	UR_1	UR_1	UR_5	UR_7	UR_9	UR_9
		$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$
	R #	0.1	0.1	0.5	0.3	0.2	0.3	0.4	0.5	0.1	0.3
	7										
	6										
	5							UR_8			
DC_2	4							UR_8	UR_8	UR_8	
	3					UR_6	UR_6	UR_6	UR_8	UR_8	
	2			UR_4	UR_4	UR_4	UR_4	UR_4	UR_4	UR_4	UR_4
	1	UR_2	UR_2	UR_4	UR_4	UR_4	UR_4	UR_4	UR_4	UR_4	UR_4
		$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$

and 3, respectively. Like overall cost, the performance of UN-FABEF, UN-RR, and UN-HAREF algorithms remains the same in the *UN* cost. Specifically, UN-FABEF performs better than UN-RR and UN-HAREF even if the *UN* time duration is assigned based on the availability of any resource slots without looking into the RE and NRE resource slots.

In the *UN* time matrix, the UN-FABEF and UN-RR perform very closely. However, the performance of UN-HAREF is poor as *UN* time is directly proportional to the RE resource slots. Alternatively, if the number of RE slots is increased, then the *UN* time is also increased, and we know that UN-HAREF uses more RE resource slots compared to other algorithms.

UN-FABEF, UN-RR, and UN-HAREF are compared using the number of used RE resources (Fig. 4). It is noticeable that UN-HAREF performs better than UN-FABEF and UN-RR as it assigns the UR to the datacenter that provides the highest RE resource slots. However, UN-FABEF also performs closely with UN-HAREF. On the contrary, UN-RR performs poorly as it circularly assigns the UR without looking into RE and NRE resource slots.

Table 10 Final Gantt chart of assigning the URs to the DCs using UN-HAREF

	R #	0.1	0.1	0.3	0.4	0.4	0.2	0.2	0.1	0.1	0.3
DC_1	7										
	6								UR_9		
	5					UR_7		UR_8	UR_9	UR_9	
	4					UR_6	UR_7	UR_8	UR_9	UR_9	
	3				$\underline{UR_5}$	UR_5	UR_6	$\underline{UR_7}$	UR_8	UR_9	UR_9
	2	$\underline{UR_3}$	$\underline{UR_3}$	$\underline{UR_3}$	$\underline{UR_3}$	$\underline{UR_3}$	UR_5	$\underline{UR_6}$	UR_8	$\underline{UR_8}$	$\underline{UR_9}$
	1	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_1}$	$\underline{UR_5}$	$\underline{UR_7}$	$\underline{UR_8}$	$\underline{UR_9}$
		$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$	$t = 7$	$t = 8$	$t = 9$	$t = 10$
	R #	0.1	0.1	0.5	0.3	0.2	0.3	0.4	0.5	0.1	0.3
DC_2	7										
	6										
	5										
	4										
	3										
	2			$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$
	1	$\underline{UR_2}$	$\underline{UR_2}$	$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$	$\underline{UR_4}$
		$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$	$t = 7$	$t = 8$	$t = 9$	$t = 10$

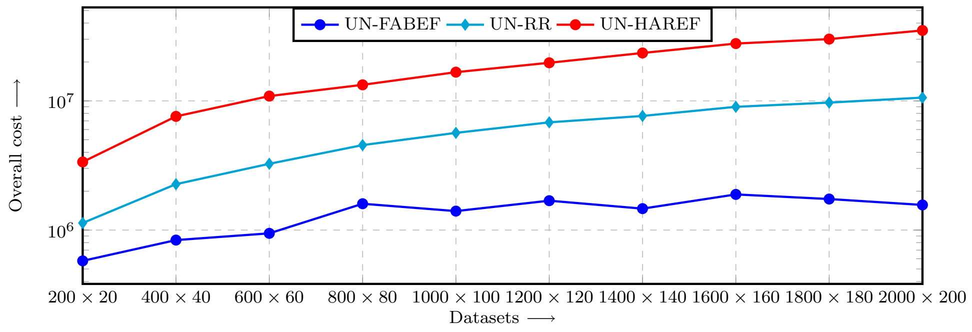

Fig. 1 Pictorial performance comparison for UN-FABEF, UN-RR and UN-HAREF algorithms in terms of overall cost

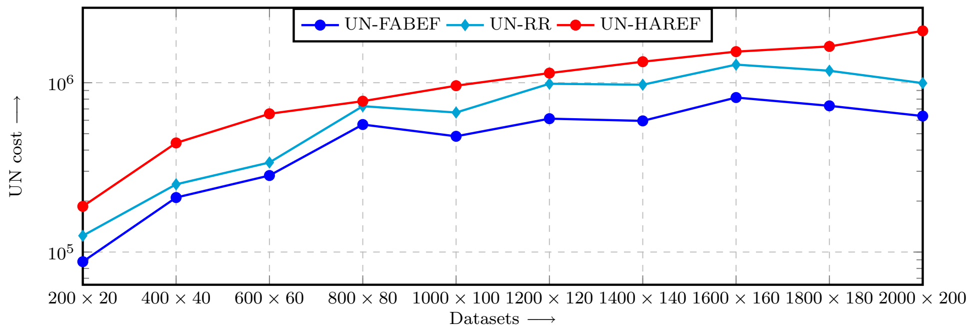

Fig. 2 Pictorial performance comparison for UN-FABEF, UN-RR and UN-HAREF algorithms in terms of UN cost

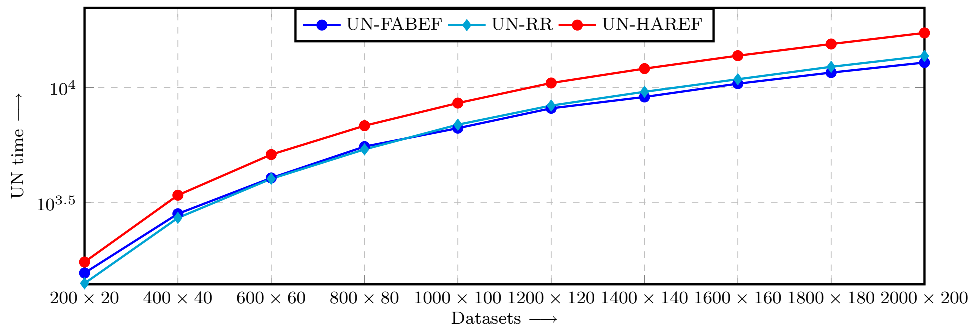

Fig. 3 Pictorial performance comparison for UN-FABEF, UN-RR and UN-HAREF algorithms in terms of UN time

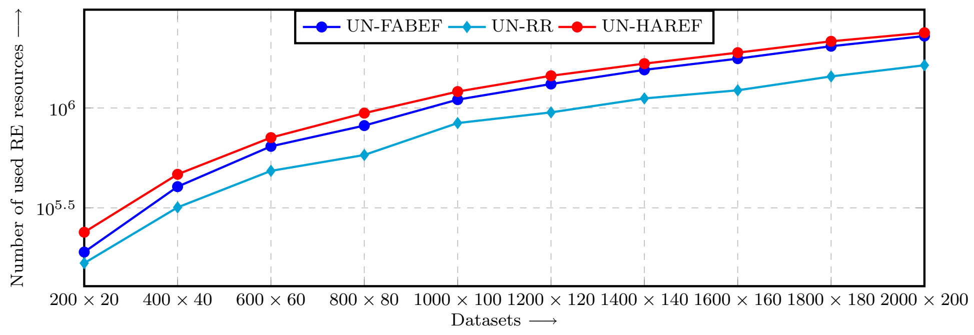

Fig. 4 Pictorial performance comparison for UN-FABEF, UN-RR and UN-HAREF algorithms in terms of number of used RE resources

6 Conclusion and future work

In this paper, we have presented three UR-based scheduling algorithms, namely UN-FABEF, UN-RR, and UN-HAREF. The aim of these algorithms is to minimize the overall cost, UN cost and UN time, and maximize the number of used RE resources. The UN is modelled in terms of the percentage of UN-RE and UN-NRE, in which UN-RE is considered as variable, and UN-NRE is fixed. Further, UN time is determined based on the number of RE and NRE resource slots. The complexity analysis of UN-FABEF, UN-RR, and UN-HAREF algorithms is performed and shown as $O(dmno)$, $O(dno)$, and $O(dmno)$, respectively. These algorithms are rigorously compared using four performance metrics by taking ten different datasets with fifty different instances. The simulation results show that UN-FABEF performs better in the overall cost, UN time, and UN cost, whereas UN-HAREF performs better in the number of used RE resources.

The proposed algorithms can be enhanced as follows.

1. UN can be modelled by considering internal and external factors. However, the detailed study of these factors is not explicitly shown in the context of scheduling algorithms.
2. UN is shown in terms of the percentage of UN-RE and UN-NRE. However, the percentage of different RE sources (i.e., solar, hydropower, wind, etc.) is not considered for simplicity.
3. UN about datacenters varies with respect to locations. However, it is not considered to avoid the complexity of the algorithms.

In our future work, we will focus on these enhancements to develop novel UR-based scheduling algorithms.

Acknowledgements None

Author contributions SP: Conceptualization, data curation, methodology, visualization, writing—original draft. RBVS: Formal analysis, investigation, methodology, validation, writing—review and editing.

Funding None.

Data availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest All the authors declared that they have no conflict of interest to disclose.

Ethical approval The manuscript "User Request-Based Scheduling Algorithms by Managing Uncertainty of Renewable Energy" meets all the ethical requirements led by the Journal.

Consent to participate This submission is guaranteed with the confirmation that the above-mentioned manuscript has not been published, accepted for publication elsewhere, or under editorial review for publication elsewhere. Furthermore, the work is not plagiarized and includes proper references wherever required.

References

1. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gen. Comput. Syst. **25**(6), 599–616 (2009)
2. Kumar, M., Sharma, S.C., Goel, A., Singh, S.P.: A comprehensive survey for scheduling techniques in cloud computing. J. Netw. Comput. Appl. **143**, 1–33 (2019)
3. Arunarani, A.R., Manjula, D., Sugumaran, V.: Task scheduling techniques in cloud computing: a literature survey. Future Gen. Comput. Syst. **91**, 407–415 (2019)
4. Wei, J., Zeng, X.: Optimal computing resource allocation algorithm in cloud computing based on hybrid differential parallel scheduling. Clust. Comput. **22**(3), 7577–7583 (2019)
5. Kumar, P., Kumar, R.: Issues and challenges of load balancing techniques in cloud computing: a survey. ACM Comput. Surv. (CSUR) **51**(6), 1–35 (2019)
6. Sumina, V..: Cloud computing statistics, facts and trends for 2022. https://www.cloudwards.net/cloud-computing-statistics/. Accessed 20 Apr 2022
7. Ochaou, L., Nacer, H., Labba, C.: Towards a distributed SaaS management system in a multi-cloudenvironment. Clust. Comput. **25**, 1–21 (2022)
8. Punitha, A., Indumathi, G.: Centralized cloud information accountability with bat key generation algorithm (CCIA-BKGA) framework in cloud computing environment. Clust. Comput. **22**(2), 3153–3164 (2019)
9. Beloglazov, A., Buyya, R., Lee, Y.C., Zomaya, A.: A taxonomy and survey of energy-efficient data centers and cloud computing systems. In: Advances in Computers, vol. 82, pp. 47–111. Elsevier, Amsterdam (2011)
10. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gen. Comput. Syst. **28**(5), 755–768 (2012)
11. Hsu, C.-H., Slagter, K.D., Chen, S.-C., Chung, Y.-C.: Optimizing energy consumption with task consolidation in clouds. Inf. Sci. **258**, 452–462 (2014)
12. Khosravi, A., Toosi, A.N., Buyya, R.: Online virtual machine migration for renewable energy usage maximization in geographically distributed cloud data centers. Concurr. Comput. Pract. Exp. **29**(18), e4125 (2017)
13. Sangaiah, A.K., Javadpour, A., Ja'fari, F., Pinto, P., Zhang, W., Balasubramanian, S.: A hybrid heuristics artificial intelligence feature selection for intrusion detection classifiers in cloud of things. Clust. Comput. **26**, 599–612 (2022)
14. Choudhary, A., Govil, M.C., Singh, G., Awasthi, L.K., Pilli, E.S.: Energy-aware scientific workflow scheduling in cloud environment. Clust. Comput. **25**, 3845–3874 (2022)
15. Uptime Institute: 2021 data center industry survey. https://uptimeinstitute.com/about-ui/press-releases/2022-global-data-center-survey-reveals-strong-industry-growth. Accessed 20 Apr 2022
16. Nayak, S., Panda, S., Das, S.: Renewable energy-based resource management in cloud computing: a review. In: Advances in Distributed Computing and Machine Learning, pp. 45–56. Springer, Singapore (2020)
17. Toosi, A.N., Buyya, R.: A fuzzy logic-based controller for cost and energy efficient load balancing in geo-distributed data centers. In: Proceedings of the 8th International Conference on Utility and Cloud Computing, pp. 186–194. IEEE Press, Piscataway (2015)
18. Nayak, S., Panda, S., Das, S.: Unconstrained power management algorithm for green cloud computing. In: Advances in Distributed Computing and Machine Learning, pp. 1–10. Springer, Singapore (2021)
19. Oberhaus, D.: Amazon, Google, Microsoft: here's who has the greenest cloud. https://www.wired.com/story/amazon-google-microsoft-green-clouds-and-hyperscale-data-centers/. Accessed 19 May 2023
20. Le, K., Bianchini, R., Zhang, J., Jaluria, Y., Meng, J., Nguyen, T.D.: Reducing electricity cost through virtual machine placement in high performance computing clouds. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, p. 22. ACM, New York (2011)
21. Chen, C., He, B., Tang, X.: Green-aware workload scheduling in geographically distributed data centers. In: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, pp. 82–89. IEEE, Piscataway (2012)
22. Panda, S.K., Jana, P.K.: Uncertainty-based QoS min–min algorithm for heterogeneous multi-cloud environment. Arab. J. Sci. Eng. **41**(8), 3003–3025 (2016)***
23. Dipu Kabir, H.M., Khosravi, A., Mondal, S.K., Rahman, M., Nahavandi, S., Buyya, R.: Uncertainty-aware decisions in cloud computing: foundations and future directions. ACM Comput. Surv. (CSUR) **54**(4), 1–30 (2021)
24. Xu, M., Buyya, R.: Managing renewable energy and carbon footprint in multi-cloud computing environments. J. Parallel Distrib. Comput. **135**, 191–202 (2020)
25. Majid, M.A., et al.: Renewable energy for sustainable development in India: current status, future prospects, challenges, employment, and investment opportunities. Energy Sustain. Soc. **10**(1), 1–36 (2020)
26. Methenitis, G., Kaisers, M., La Poutré, H.: Renewable electricity trading through slas. Energy Inf. **1**(1), 1–17 (2018)
27. Ahmed, I.: Sustainable Green Service Level Agreement (GSLA) framework development for IT and ICT based industries. PhD thesis, Saga University, Saga, Japan (2018)
28. Koutsoyiannis, D.: The unavoidable uncertainty of renewable energy and its management. In: Proceedings of the EGU General Assembly Conference, Vienna, Austria, pp. 17–22 (2016)
29. de Carvalho, P.S., Siluk, J.C.M., Schaefer, J.L., et al.: Analysis of factors that interfere with the regulatory energy process with emphasis on the energy cloud. Int. J. Energy Econ. Policy **12**(2), 325–335 (2022)
30. Panda, S.K., Jana, P.K.: An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems. Clust. Comput. **22**(2), 509–527 (2019)
31. Nayak, S., Panda, S., Das, S., Pande, S.: A multi-objective renewable energy-based algorithm for geographically distributed datacentres. Int. J. Embed. Syst. **15**(2), 119–131 (2022)

Slokashree Padhi is working as a Ph.D. research scholar in the Department of CSE at NIT Warangal, Telangana, India. She worked as an Assistant Professor in the Department of CSE at SIT Sambalpur, Odisha, India. She received M. Tech. degree from VSSUT Burla, Odisha, India and B. Tech. degree from GCEK, Bhawanipatna, Odisha, India, in CSE. Her current research interests include cloud computing, load balancing, renewable energy scheduling and machine learning.

R. B. V. Subramanyam is working as a Professor in the Department of CSE at NIT Warangal, Telangana, India and Chief Investigator, E & ICT Academy under Ministry of Electronics and Information Technology (MeitY), Govt. of India. He received Ph.D. degree and M. Tech. degree from IIT Kharagpur, West Bengal, India. He received many awards from reputed organizations, including IEEE senior member. His current research interests include cloud computing, machine learning, data mining, distributed data mining, graph databases, fuzzy data mining, big data analytics, cluster computing, pattern recognition, high performance computing, soft computing and outlier analysis. He has published more than 40 papers in reputed journals and conferences, including Expert Systems and Applications, Elsevier, Engineering Science and Technology, Elsevier, Computer and Information Sciences, Elsevier, Information and Computer Science Journal, Elsevier, Cluster Computing, Springer, Applied Intelligence, Springer, Geoinformatica, International Journal of Data Analysis Techniques and Strategies, Inderscience, Journal of Information and Knowledge Management, Enterprise Information Systems and many more. He has awarded more than 8 Ph.D. scholars. He has completed three projects from MHRD, Govt. of India and Infosys Limited. He delivered many invited talks, and chaired sessions in many national and international conferences. He acted as reviewers in many reputed journals of IEEE, Elsevier and Springer and many reputed conferences.