The Journal of Supercomputing (2023) 79:8153–8177
https://doi.org/10.1007/s11227-022-04690-2

Multi-objective approach for scheduling time-aware business processes in cloud-fog environment

Fairouz Fakhfakh[1,2] · Saoussen Cheikhrouhou[1] · Bouthaina Dammak[3] · Monia Hamdi[4] · Mouna Rekik[5]

Accepted: 22 June 2022 / Published online: 23 July 2022

Abstract

Currently, constant innovations in service-oriented architectures lead to extending the Cloud services with promising solutions such as Fog computing. As the Cloud-Fog environment still remains in its infancy stage, several issues remain among the considerable challenges to be handled. One of the key issues in a Cloud-Fog environment is the scheduling of business processes tasks, i. e. selecting the suitable Cloud-Fog resources to support the execution of the business processes tasks while considering budget and temporal constraints. Indeed, these constraints are generally contradictory. Indeed, the use of cheaper resources increases the execution time and vice versa. Furthermore, minimizing the energy consumption is among the prominent considerations when dealing with Cloud-Fog environment. Hence, finding out the trade-off set of optimal solutions is required considering minimizing cost, time and energy consumption. To address such an issue, we propose, in this paper, a Multi-Objectives Particle Swarm Optimization (MOPSO) algorithm based on a non-dominance sort to handle the scheduling problem of time-aware business processes with many conflicting objective functions. Our algorithm aims to optimize three conflicting objectives namely, the makespan (total execution time), the monetary cost and the energy consumption while taking into account budget and temporal constraints of the business process. The output of our MOPSO algorithm represents a set of Pareto optimal solutions from which the user can select the best one. The elaborated experimentation illustrates the good performance of the proposed algorithm.

Keywords Business process · Scheduling · Cloud-Fog environment · Temporal constraints · Multiple objectives · Particle swarm optimization

✉ Bouthaina Dammak
BAdammak@pnu.edu.sa

Extended author information available on the last page of the article

1 Introduction

During the last decade, Cloud computing technology has witnessed an exponential interest and an explosion of its popularity regarding the various kind of services it offers [1–3]. This expansion is motivated by the ever-increasing use of the Internet of Things systems (IoT) that require access to the data storage from any geographical location with an internet connection [4, 5]. Despite the great number of Cloud benefits and advantages, the centralized nature of this paradigm leads to some shortages [6]. The large data volumes transmitted from IoT devices, whose number is will be forecast to more than 25.4 billion by 2030, cause network congestion. Moreover, cloud data centers are physically distant away from data sources so that data transmission takes too long time to reach the Cloud. Consequently, the cost of Cloud resources consumption as well as the bandwidth use is getting more and more expensive. Meanwhile, recent applications have been exposed to real time constraints. Therefore, a computing paradigm convenient to tackle the issue of variety, rapidity, and volume of the data generated by IoT devices is needed.

The adoption of the Fog as an extension of the Cloud environment is among the proposed solutions. Fog computing is located basically between the Internet of Things (IoT) devices and the Cloud servers. Furthermore, the Fog computing is characterized by its distributed nature and it is constituted mainly by devices that are closer to the end users terminals [7]. More specifically, the Fog devices are supposed to handle a considerable amount of service request processing steps such as data filtering and analysis before being transmitted to the Cloud resources. Fog computing has become a popular paradigm, which provides computing resources for end-users' applications [8]. It can solve several shortages of Cloud Computing based systems such as the poor support of the user's mobility, unexpected delays and heavy communication [9].

Due to the different types of Cloud and Fog computing resources, efficient assignment of tasks to resources has become a major challenge. It is difficult to make an appropriate decision when mapping tasks to resource types considering multiple objectives that are often contradictory. This problem has become complex for business processes which impose dependencies and order constraints between tasks. In that respect, some works have focused on optimizing the timespan or resource consumption on Fog-Cloud systems. In some other papers, researchers have focused on finding a balance between resource cost or time span and energy consumption. In [10–15], authors have only considered the traditional objectives, such as resource cost and makespan (total execution time). However, to the best of our knowledge, previous works have not yet handled three objectives at the same time under deadline and budget constraints. Also, few authors have paid attention to temporal constraints associated with tasks during scheduling business processes [16, 17]. In our previous work [18], we have addressed the scheduling problem for timed-constrained business processes while taking into account only one objective (resource cost reduction). To do so, we have compared two algorithms based on two optimization methods: the exact method and the meta-heuristic method. The first one adopts an exact optimization method which provides an optimal assignment of tasks to resources. However, the second solution is based on the meta-heuristic particle swarm optimization (PSO) which generates a good solution with no guarantee of optimality, but with the benefit of a shorter computation time. The main shortcoming of [18] is the consideration of only the resource cost issue. A scheduler that minimizes the resource of an application can have a long makespan or may consume considerable energy which are very important metrics in Fog-Cloud computing [19, 20]. Indeed, Fog-computing devices are energy constrained, since they are mainly battery powered and cannot be connected to the main electric grid. This is the case of wide areas desert or polar marines where computation is needed to process data coming from sensors and drones [19].

Our contributions consist of the following :

1. Using the multi-objective optimization approach to generate Pareto optimal solutions for our scheduling problem. We propose a multi-objective Particle Swarm Optimization (MPSO) algorithm for time-aware business processes in Cloud-Fog environment based upon a non-dominance sorting procedure.
2. We consider two conflicting objectives (the makespan and the total resource cost), as well as the energy consumed by the scheduler.
3. The MPSO algorithm takes into account crucial parameters; the deadline, the budget and the duration constraints of tasks.

The simulation experiments show that our algorithm delivers better results in terms of computation time, cost and makespan trade-off as well as the respect of deadline and budget constraints.

The remainder of this paper is organized as follows: In Sect. 2, we show a motivating scenario based on the collaboration between Cloud and Fog computing. Sect. 3 describes the assumptions considered on application and resource levels. In Sect. 4, we briefly introduce the approach of multi-objective optimization and PSO algorithm. In addition, we detail our proposed algorithm. In Sect. 5, we describe the experimental settings of our evaluation and we discuss the experimental results. A review of the related literature is presented in Sect. 6. Finally, the last section concludes the paper and outlines some future works.

2 Motivation scenario

Nowadays, organisations embrace time-based competition as a strategy to gain competitive advantage in today's economy. This strategy focuses on time management, especially delivery lead time, to quickly respond to customers' requirements. Respecting customers' promised delivery time (i.e., process deadline) is crucial to attract more customers. Consequently, respecting all the orders on time (i.e., respecting process deadlines) is harder given the increased number of customer orders, uncertainties in available resources and delivery processes, or other unpredictable

Table 1 Tasks of drone shipping business-process

Task	Description
T_1	Collect and verify delivery instructions
T_2	Start pick process by robot-smart warehouse
T_3	Sense drone position
T_4	Receive current position and further information sent by drone
T_5	Maintain a logbook of drone flying/delivery activities
T_6	Process received information
T_7	Send instructions to avoid accidents or violations
T_8	Update drone position
T_9	Notify the customer about product arrival

events. Otherwise, organisations will assume the poor on-time delivery and will be charged with higher tardiness cost, and thereby the firm's long-term competitive advantage is threatened.

In this work, we assume that controlling the execution of process tasks in Cloud and/or Fog will have significant effects on avoiding the late deliveries of processes (i.e, avoiding deadline violations).

For example, the online retailer service of Amazon is not only sensitive to the selling price, but also to the promised delivery lead time.

Indeed, it launched the Amazon's Prime Air project,[1] in which the firm revealed its somewhat audacious plan to make deliveries by drone. On December 7th, 2016, the firm announced the completion of the first Amazon Prime Air delivery, with a shipment lasting only 13 minutes from order to delivery.

When ordering goods that weigh less than 2.6kg, and being close to an Amazon depot, Amazon can offer the Prime Air drone delivery service. In this case, the process needs to migrate some tasks from Cloud to Fog to respond to the drone real-time needs, reaching thus the specified *lead time*.

To illustrate the features of the proposed approach, we introduce a toy process to exemplify the drone delivery process of Amazon. The process starts by collecting delivery instructions and starts the pick up process. It tracks the drone with sensing its position, receiving its current position, maintaining a logbook of drone flying/delivery activities. Also, this process handles received information, sends instructions to avoid accidents or violations, and updates drone position. Finally, it notifies the customer about the product arrival. Each task of this time-aware business process has a temporal duration constraint defined by the designer.

The combination of a huge network of warehouses, excellent transportation, and most importantly the use of information technology such as Cloud and Fog computing, makes Amazon online retailer the most efficient compared to major companies in the world. In the case of Amazon's local distribution system, distribution tasks may benefit from both Cloud and Fog computing capabilities to ensure

[1] https://www.engadget.com/2016/12/14/amazon-completes-its-first-drone-powered-delivery/.

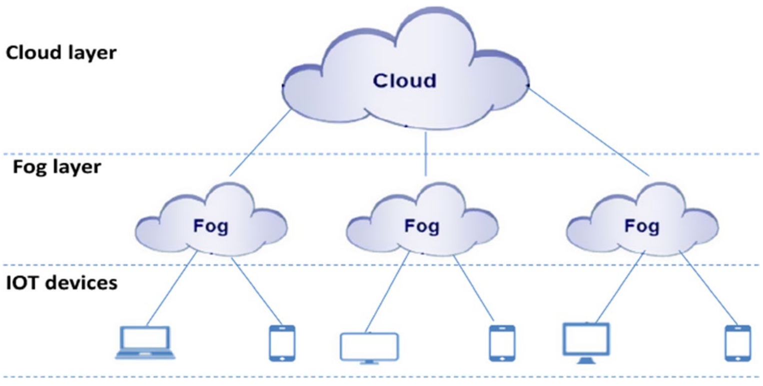

Fig. 1 Cloud and Fog computing architecture

rapid fulfillment to consumers. In this work, we consider M types of computational resources R $=\{R_1, R_2,..., R_M\}$ which consist of Cloud and Fog resources. The description of the process tasks are presented in Table 1. Each task can be assigned to one of the available resource types. The mapping result can only be determined after applying our scheduling algorithm that will be introduced in Sect. 4.

3 Problem formulation

In this section, we introduce the main concepts and assumptions related to our application, resources and scheduling model. Our goal is to assign the tasks of a time-aware business process to Cloud and Fog resources in order to optimize three conflicting objectives while satisfying a set of constraints. Our Cloud-Fog computing system consists of three layers in a hierarchical network (see Fig. 1). The bottom layer includes IoT devices, which are used as users interface that transmit requests from users. The middle layer is formed by a set of Fog nodes with computing, storage and network capabilities. The different Fog nodes receive the users' requests, process them and transmit results to the upper layer (Cloud layer). The upper layer hosts a set of heterogeneous Cloud resources of a Cloud service provider.

3.1 Application model

A time-aware business process model is represented by a directed acyclic graph (DAG), denoted G = (T,E), where T = $\{T_1, T_2,..., T_N\}$ is the task set and E is the edge set defining the precedence constraints between tasks (see Fig. 2). N denotes the number of tasks of the business process. The task T_i is identified by its index i. An edge $e_{i,j} = (T_i, T_j) \in$ E means that T_j cannot be executed until T_i is completed. As shown in Fig. 2, task T_8 can only be executed when all of its predecessors (T_3, T_5 and T_7) have been executed. However, parallel tasks of the business process can be executed concurrently. For example, T_2, T_4 and T_6 are three concurrent tasks that can be executed at the same time. The task T_i without any predecessor is denoted T_{start}

Table 2 Descriptions of Symbols

Symbols	Descriptions
G	The directed acyclic graph
T	The set of tasks in a business process
N	The number of tasks of the business process
T_i	The task identified by its index i
E	The edge set defining the precedence constraints between tasks
$e_{i,j} = (T_i, T_j)$	The task T_j cannot be executed until T_i is completed
T_{start}	The beginning task of the graph
T_{end}	The end task of the graph
$w_{i,j}$	The weight value of the edge $e_{i,j}$
DC_i	The duration constraint of the task T_i
D	The deadline constraint of the business process

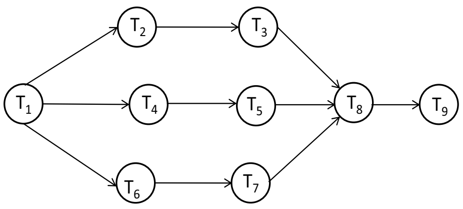

Fig. 2 An illustrative DAG

and the task T_j without any successor is denoted T_{end}. Each edge $e_{i,j}$ has its weight value $w_{i,j}$ ($w_{i,j} \in \mathbf{R}+$) which designates the amount of data that must be transferred from T_i to T_j. As our scheduling algorithm requires a single start and a single exit task, we always add two dummy tasks (T_{start} and T_{end}) to the beginning and the end of the graph, respectively. These dummy tasks do not require any resource. They have zero execution time and they are connected with zero weight to the actual start and exit tasks.

In our work, each process has a deadline constraint (D) which is defined as the time constraint imposed by the user on a process. Deadline is defined as the time constraint during which all process tasks must complete their execution. The designer can specify whether this constraint is critical depending on the application. Also, we define temporal constraints associated with tasks to limit their execution time on the processing node. We notice, DC_i the duration constraint of the task T_i. Table 2 provides a summary of the symbols used in this paper.

3.2 Resources model

The resources model can be expressed as a set of M computational resources R $=\{R_1, R_2,..., R_M\}$ which consists of Cloud and Fog resources. Let R_c and R_f denote the set of Cloud resources and the set of Fog resources, respectively. Hence, R $=R_c$ U R_f. Each computational resource type R_t (t $\in \{1, 2, ..., M\}$) is characterized by its price per unit of time denoted C_t. We assume that each resource can only execute one task at the same time. In addition, we adopt the hourly pricing model. Then, resources are purchased according to the number of usage hours.

- The total execution time of a business process consists of two parts: the execution time of tasks and the communication time.

  (i) The execution time of the task T_i running either on a Cloud or Fog resource R_t is denoted as d_i^t ($t \in \{1..M\}$). T_i must satisfy its duration constraint DC_i.

$$d_i^t \leq \mathrm{DC}_i \tag{1}$$

  (ii) The communication time between the two successive tasks T_i and T_j is defined as follows:

$$\mathrm{CT}_{ij} = \frac{w_{i,j}}{\mathrm{BW}} \tag{2}$$

  where BW represents the bandwidth between the resources assigned respectively to T_i and T_j and ($w_{i,j} \in \mathbf{R}+$) is the amount of communication data transferred between these two tasks. The resources in the Fog and Cloud layers are fully connected by a virtual network that connects the two layers over the Internet. The bandwidth between the two Fog resources R_i^{fog} and R_j^{fog} is marked by $\mathrm{BW}_{ij}^{\mathrm{fog}}$. On the other hand, the bandwidth between the two Cloud resources R_i^{cloud} and R_j^{cloud} is marked by $\mathrm{BW}_{ij}^{\mathrm{cloud}}$. Finally, the bandwidth between a Fog resource R_i^{fog} and a Cloud resource R_j^{cloud} is marked by $\mathrm{BW}_{ij}^{\mathrm{inter}}$. It is noted that the superscript indicators in the names of variables are used to distinguish between the variables appropriate to each layer.

- The Start Execution Time (SET_i) and the Finish Execution Time (FET_i) of the task T_i running on the resource R_t are defined by the Eqs. (3) and (4).

$$SET_i = \begin{cases} 0 & \text{if } T_i = T_{start} \\ max\{\mathrm{FET}_j + d_i^t + \mathrm{CT}_{ji}\}, (T_j, T_i) \in E & \text{otherwise} \end{cases} \tag{3}$$

  We assume that the start execution time of the first task T_{start} is equal to zero. Otherwise, it represents the maximum of the sum of the finish execution time of all the preceding tasks T_j of T_i, the execution time (duration) of T_i and the communication time between T_j and T_i. The finish execution time of a task T_i is computed as the sum of its start execution time and its execution time.

$$\mathrm{FET}_i = \mathrm{SET}_i + d_i^t \tag{4}$$

- The execution cost of the task T_i running on the resource R_t is denoted as C_i^t. It is defined as the product of the number of the required hours to execute T_i on R_t (Nbh_i^t) by the unit price of R_t (C_t).

$$C_i^t = \mathrm{Nbh}_i^t \times C_t \tag{5}$$

3.3 Scheduling model

This work aims to find a schedule to execute a time-aware business process on Cloud and Fog computing resources such that the overall makespan, execution cost and consumed energy are minimized while respecting a set of constraints. In what follows, we explain the objectives and the constraints of our scheduling model.

3.3.1 Makespan

The first objective Obj1 is to reduce the makespan in such a manner that each task is executed before its duration constraint and the process completes its execution before the deadline D.

$$Obj1 = \mathrm{Min}(\mathrm{makespan}) \tag{6}$$

$$\mathrm{makespan} \le D \tag{7}$$

The makespan is defined as the maximum of the finish execution time of all business process tasks as follows:

$$\mathrm{makespan} = \mathrm{Max}(\mathrm{FET}_i), i \in \{1, .., N\} \tag{8}$$

3.3.2 Execution cost

The total cost of resources required to run a process is the sum of resource costs of all the tasks. The objective *Obj*2 is to execute a process with least amount of money so that the total resource cost does not exceed the specified user budget. The user budget represents the financial cost specified by the user to execute its business process.

$$\mathrm{Obj2} = \mathrm{Min}\left(\sum_{i=1}^{N} (C_i^t)\right) \tag{9}$$

$$\sum_{i=1}^{N} (C_i^t) \le B \tag{10}$$

3.3.3 Energy consumption

We estimate the energy consumed by a business process based on its power consumption and its execution time on the resource. The power consumption is divided into dynamic and static components. According to [14, 21], the dynamic power

($P_{dynamic}$) is the dominant and expensive component of energy while the static power (P_{static}), consumed by the different resources when the system does not execute any workload, has relatively smaller value.

In our model, the power consumption is represented as:

$$P = P_{\text{dynamic}} + P_{\text{static}} \tag{11}$$

To execute the tasks of a business process in an energy efficient manner, we have adopted Dynamic Voltage Frequency Scaling (DVFS). In this case, the dynamic power is computed as [22]:

$$P_{\text{dynamic}} = A \times C \times V^2 \times f \tag{12}$$

where A is the number of switches per clock cycle, C is the total capacitance load, V is the supply voltage and f is the frequency.

The power consumed by the task T_i on the resource R_t is calculated as :

$$P(T_i, R_t) = A \times C \times V_i^2 \times f_i + P_{\text{static}} \tag{13}$$

V_i, f_i are the voltage supply and frequency of the processor on which task T_i is executed.

The energy consumed during the execution of a task T_i on the resource R_t is calculated as [22]:

$$\text{EC}(T_i, R_t) = P(T_i, R_t) \times d_i^t \tag{14}$$

where V_i is the supply voltage of the resource R_t (processor) on which task T_i is executed and d_i^t is the execution time of task T_i on the resource R_t.

The energy consumed by executing business process tasks over available resources is given by:

$$\text{Energy} = \sum_{i=1}^{N} \text{EC}(T_i, R_t) \tag{15}$$

The objective Obj3 is to execute a business process with minimum energy; this can be achieved by reducing the sum of energy consumed by the process tasks as follows:

$$\text{Obj3} = \text{Min}(\text{Energy}) \tag{16}$$

4 Scheduling time-aware business process tasks

In this section, we present an overview on multi-objective combinatorial optimization and the Particle Swarm Optimization algorithm. After that, we detail our proposed algorithm to solve the multi-objective problem of scheduling time-aware business processes in a Cloud-Fog environment.

4.1 Multi-objective optimization

The Multi-objective Optimization aims to simultaneously optimize multiple conflicting objectives by minimizing or maximizing them [23]. A Multi-objective Optimization Problem (MOP) [24] with m decision variables and k objectives can be formally defined as follows:

$$\mathrm{Min}(f(x)) = \mathrm{Min}(f_1(x), f_2(x), ..., f_k(x)) \tag{17}$$

where:

- $f_1(x), f_2(x), ..., f_k(x)$ are k different conflicting objectives
- $\mathrm{X} = (X_1, X_2, ..., X_m)$ is the search space of the problem.

In MOP, there is no single optimal solution since multiple conflicting objectives are implied. In fact, it is not possible to determine whether a solution is better than another. So, the concept of Pareto Dominance is adopted to compare the solutions. Some of the Pareto concepts [25] used in MOP are presented in what follows:

Pareto dominance For two decision vectors x_1 and x_2, dominance (denoted by $\prec$) is defined as follows: The decision vector x_1 is said to dominate x_2 if and only if, x_1 is as good as x_2 for all the objectives and x_1 is strictly superior to x_2 in at least one objective.

$$x_1 \succ x_2 \Leftrightarrow \forall f_i(x_1) \leq f_i(x_2) \wedge \exists j f_j(x_1) < f_j(x_2) \tag{18}$$

Pareto optimal set The Pareto optimal set P_s is the set of all Pareto optimal decision vectors. It is defined as: where the decision vector, x_1, is said to be Pareto optimal when it is not dominated by any other decision vectors, x_2, in the set.

$$P_s = \{x_1 \in X, | \nexists x_2 \in X, x_2 \prec x_1\} \tag{19}$$

Pareto optimal front The Pareto optimal front P_f is defined as the image of the Pareto optimal set in the objective space.

$$P_f = \{f(x) = (f_1(x), ..., f_n(x)) | x \in P_s\} \tag{20}$$

4.2 Particle swarm optimization

PSO is a swarm-based intelligence algorithm [26] derived from the behavior of animals such as a flock of birds looking for a source of food. The main idea of PSO consists in searching an optimization solution by sharing information between the members of a group which is commonly called “population”. PSO begins with a set of potential solutions which are initialized randomly. Each

Table 3 Mapping between tasks and resources types

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
Resource type	R_3	R_1	R_2	R_1	R_3	R_4	R_4	R_2	R_5

Table 4 The particle corresponding to the business process

3	1	2	1	3	4	4	2	5

solution, called particle, has two parameters that are position and velocity. The position of a particle P_i is affected by its optimal position denoted as *pbest* and the best position of the whole population denoted as *gbest*. The performance of a particle X is evaluated based on the value of its fitness function F(X) which is specific to the studied problem. At an iteration *k* of the algorithm, the position and the velocity of a particle are updated according to the following equations:

Updating Position Vector

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{21}$$

where x_i^k and x_i^{k+1} indicate respectively the current position of a particle P_i at the iteration k and k+1.

Updating Velocity Vector

$$v_i^{k+1} = w \times v_i^k + c_1 \times r_1 \times (pbest_i - x_i^k) + c_2 \times r_2 \times (gbest - x_i^k) \tag{22}$$

where $c1$ and $c2$ designate the acceleration factors, $r1$ and $r2$ are two random numbers within [0, 1] and w represents the inertia weight. $pbest_i$ is the best position of the particle P_i and *gbest* represents the position of the best particle in a population.

The choice of PSO algorithm, in our work, is justified by its fast convergence and its fewer parameters to tune [27]. It has no overlapping and mutation calculation like genetic algorithms. In addition, at each iteration, only the most optimistic particle can transmit information onto the other particles. Furthermore, PSO has obtained a great success in dealing with the scheduling problems [28–30].

In order to provide a mapping solution of all process tasks to the appropriate resource types, we represent a particle as a process and its resources. The dimension of a particle is defined as the number of the process tasks. The range in which a particle can be positioned is determined by the indexes of the available resource types. Table 3 shows an example of a mapping between the tasks of our motivating example and the resource types (We consider 6 resource types R ={R_1, R_2,..., R_6}). Thus, the corresponding particle can be seen in Table 4. The value of this particle in each dimension represents the index of the resource type.

4.3 Proposed algorithm

In order to solve the multi-objective problem of scheduling time-aware business processes, we have proposed a Multi-Objective Particle Swarm Optimization (MOPSO) algorithm. The latter is based on a dominance sorting procedure. Our scheduling problem is based on a fitness function $F(X)$ which consists of three objectives.

$$F(X) = W_1 \times \mathrm{obj1} + W_2 \times obj2 + W_3 \times obj3 \tag{23}$$

where obj1, obj2 and obj3 are the objective functions already defined in Eqs. (4), (5) and (13).

In such a method, the weights W_s $(0 \leq W_s \leq 1)$ of objectives are dynamic weights which are iteratively adjusted by the algorithm. They usually decrease from a large value (e.g., 0.9) in the beginning to a small value (e.g., 0.1) in the final iteration to control the exploration of possible solutions.

During each iteration of the algorithm, a schedule solution of a particle may be unfeasible due to the constraints violation. There are two types of constraints: the temporal constraints that impose temporal behavior of tasks and the deadline of business process, and a budget constraint. A particle solution is considered as feasible if it satisfies all these constraints. Otherwise, it is called unfeasible. Thus, the idea is to give a penalty to the fitness values of unfeasible solutions of particles. So, the fitness values of unfeasible solutions will be reduced compared to those of feasible solutions. This process can reinforce the presence of the feasible solutions. Equation (24) shows the definition of our fitness function.

$$\mathrm{Fitness}(X) = \begin{cases} F(X) \times 0.5 + 0.5 & \text{if there is no constraints violation} \\ F(X) \times 0.5 + 0.1 \times P(X) & \text{otherwise.} \end{cases} \tag{24}$$

$$P(X) = \sum_{i=1}^{N} \frac{\mathrm{DC}i(X) \times v_i(X)}{d_i(X)} + \frac{D \times v_D(X)}{\mathrm{makespan(X)}} + \frac{B \times v_B(X)}{\mathrm{cost(X)}} \tag{25}$$

where:

- $v_i(X) = 1$ if the task T_i $(i \in \{1, .., N\})$ of the particle X does not satisfy its duration constraint, otherwise, it takes 0.
- $DC\ i(X)$: is the duration constraint of the task T_i in the particle X
- $d_i(X)$: is the duration of the task T_i in the particle X
- $v_D(X) = 1$ if the particle X does not satisfy its deadline constraint, otherwise, it takes 0.
- $v_B(X) = 1$ if the particle X does not satisfy its budget constraint, otherwise, it takes 0.
- $makespan(X)$ is the total execution time of the particle X
- $cost(X)$ is the financial cost of the resources used for the particle X
- D and B represent, respectively, the deadline and budget constraints of the business process

In order to determine the penalty function (equation 25), we have relied on some existing papers in the literature [31, 32]. Each part of this function takes into account a constraint. More specifically, we consider three constraints that we transform to penalties. The first part of this function is intended for the duration constraint of each task. The second part deals with the deadline constraint of the business process. Finally, the third part deals with the budget constraint of the resources cost. In our proposed algorithm, the more a particle violates the defined constraints, the more it is penalized.

The details of our scheduling algorithm are described in Algorithm 1.

Algorithm 1 Multi-objective scheduling algorithm for a time-aware business process

```
1: Input: A business process (BP) with temporal constraints, budget constraint (B), number of iterations ($nb\_max\_iter$), a set of Cloud and fog resources
2: Output: Non-dominant solutions of BP scheduling
3: Set iteration counter k=0
4: Set particle dimension as equal to the size of the process tasks and initialize population particles by randomly assigning the BP tasks to the resource types.
5: Calculate the fitness values of each particle according to the equation (24)
6: Insert the Pareto optimal (non-dominated) solutions into an archive A
7: while maximum iteration number hasn't been reached do
8:     for each particle $P_i$ with position $x_i$ do
9:         Sort archive members based on $Fitness(x_i)$
10:        if $Fitness(x_i)$ is better than $Fitness(pbest_i)$ then
11:            $pbest_i \longleftarrow x_i$
12:        end if
13:        Calculate the position and the velocity of the particle according to the equations (21) and (22).
14:    end for
15:    $gbest_i \longleftarrow$ The position of the best particle
16:    Set k=k+1
17:    if k < $nb\_max\_iter$ then
18:        Update the archive contents by deleting the dominated members and storing the Pareto optimal solutions.
19:    end if
20: end while
21: Output the Pareto optimal solutions from the Archive A
```

Algorithm 1 starts by initializing the iteration number to 0 (line 3). In addition, it sets the particle dimension equal to the number of process tasks and it initializes the particles by randomly assigning the tasks to the resource types (line 4). After that, the values of the fitness functions are computed (line 5).

During each iteration of the algorithm, the particles stored in the archive are sorted in an ascending order according to the value of the total fitness function *Fitness* (line 9). Also, their corresponding *pbest* are updated (line 11). Then, new velocity and position of each particle are computed (line 13) and the position of the best particle *gbest* is determined (line 15).

The archive content is updated in every iteration (line 18). Once the maximum iteration number is reached, the archive containing the Pareto optimal solutions is returned accordingly (line 21).

In order to convert the particle's position into a schedule, we introduce the pseudo code of Algorithm 2. This algorithm takes a set of inputs:

- T: is the set of business process tasks
- R: is the set of available resources
- P: is a particle which represents a mapping of tasks to resources
- d[T× R]: is a matrix in which the rows represent the tasks and the columns represent the available resources. It contains the execution time of all tasks on the different resources *R*.
- CT[T× T] : is a matrix which contains the communication time between tasks.

The computaion of the start execution time values of business process tasks is based on Eq. 3, which assumes that the first task T_{start} begins its execution at t = 0 (line 3). After that, a task T_i can start running as soon as their predecessors (parents) tasks complete their execution and the output data is transferred (line 6).

Algorithm 2 Schedule Generation

```
1: Input: T, R, P, d[T× R], CT[T× T]
2: Output: The start execution time and the finish execution time of tasks
3: SET($T_{start}$)=0
4: for each task $T_i \in T \setminus \{T_{start}\}$ do
5:     $FET_i = SET_i + d_i^t$
6:     $SET_i = max\{FET_j + d_i^t + CT_{ji}\ \}$, $(T_j, T_i) \in$ E
7: end for
```

5 Evaluation

In this section, we describe the experimental settings of our time-aware business processes, resources and algorithms. After that, we present the simulation experiments[2] that we have conducted to evaluate the efficiency of the proposed algorithms.

5.1 Experimental setup

The different experiments were performed on a laptop that has 64-bit Intel Core, 2.6 GHz CPU, 8 Go of RAM and Windows 7 as OS. The experimental results are based on a dataset that consists of a set of randomly generated business processes with different sizes, this will be presented in next subsection. Concerning the tasks of business processes, we generated a set of data randomly. The size of data transferred from a task to another is ranging from 200 to 1600 MB.

[2] The datasets used during the current study are available from the corresponding author on reasonable request.

For implementing our algorithm based on PSO, the population size (number of particles) is set as 50 and the dimension of each particle is equal to the number of process tasks. We define the acceleration factors $c1 = c2 = 2$ and the inertia value $w = 0.5$ (These parameters are already explained in Sect. 4.2). The number of iterations needed for the convergence depends on the number of tasks of the used business process.

In order to determine the operation levels for DVFS, we have relied on the work of Yassa et al. [33] which considers that each resource $R_i \in R$ is DVFS-enabled. In other words, it can operate on various voltage scaling levels (i.e., at different clock frequencies). For each resource $R_i \in R$, a set of voltage-scaling levels is randomly and uniformly distributed among the three sets of voltage-scaling levels shown in Table 5. After determining the number of voltage levels supported by the different resources, relative speed and frequency of the resources are determined. We assume that resources consume energy during the periods of inactivity. In other words, when a resource is idle, it is assumed that the lowest voltage is supplied [34]. For computing the energy of created schedule, we use Equation (15) as detailed in Sect. 3.

In order to evaluate the performance of our algorithm, we generate the values of the Budget B and the Deadline D constraints as follows:

$$BudgetB = Min_B + k1 \times (Max_B - Min_B) \tag{26}$$

where Min_B is the minimum cost obtained by mapping each task to the cheapest resource, Max_B is the maximum cost obtained conversely and $k1$ is a budget ratio in ranging from 0 to 1.

$$DeadlineD = Min_D + k2 \times (Max_D - Min_D) \tag{27}$$

where Min_D and Max_D represent respectively the minimum and maximum makespan obtained by mapping time-aware business process tasks to resources. $k2$ is a deadline ratio ranging from 0 to 1.

5.2 Experimental results

This section shows simulation results and analysis of our proposed algorithm. To measure the efficiency of our algorithm, we compare its results with two other existing algorithms called Non-dominated Sort Genetic Algorithm (NSGA-II) [35] and SPEA2 (Strength Pareto Evolutionary Algorithm) [36], which are adapted and simulated to handle the multi-objective scheduling problem for time-constrained processes in a Cloud-Fog environment. In each experiment, we execute each scenario 100 times while using different sizes of data and we report the average value of these executions.

5.2.1 Evaluation of the computation time

In this experiment, we report the execution time required to schedule time-aware business process tasks to resources using the three mentioned algorithms. We present our evaluation in the case of two scenarios: increasing the number of tasks and

Table 5 Voltage-Relative Speed Pairs of resources [33]

Level	R1			R2			R3		
	Voltage (V)	Freq (Ghz)	Speed (%)	Voltage (V)	Freq (Ghz)	Speed (%)	Voltage (V)	Freq (Ghz)	Speed (%)
1	1.2	1.8	100	1.35	0.6	100	1.48	1.4	100
2	1.15	1.6	88.88	1.27	0.55	91.66	1.44	1.2	96.76
3	1.1	1.4	77.77	1.2	0.5	83.33	1.31	1	88.14
4	1.05	1.2	66.66	1	0.25	41.66	1.18	0.8	79.51
5	1	1	55.555	0.9	0.13	20.83	0.96	0.6	64.42
6	0.9	0.8	44.44						

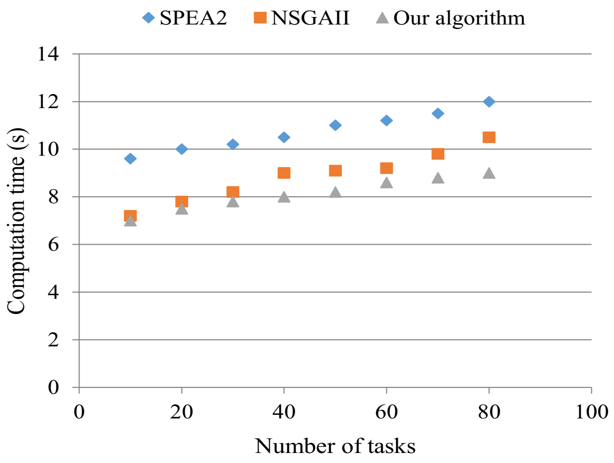

Fig. 3 Evaluation of the computation time while increasing the number of tasks

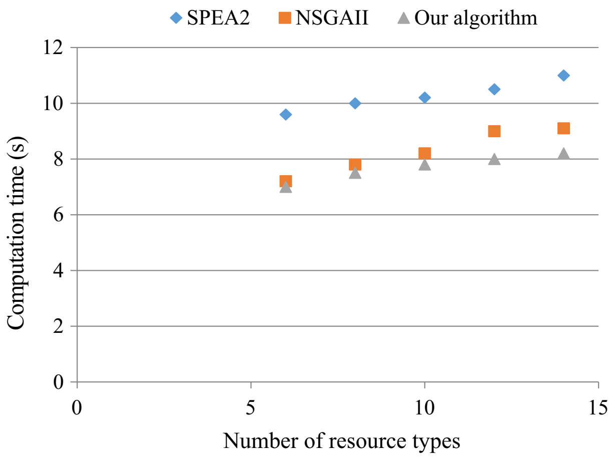

Fig. 4 Evaluation of the computation time while increasing the number of resource types

that of resource types. The results are depicted respectively in Fig. 3 and Fig. 4. We notice that, when the number of tasks and resources types are small, the computation time is very low using NSGAII and our algorithm, comparing it with the computation time of SPEA2 algorithm. However, when the number of tasks and resource types become considerable, our algorithm is the most efficient one. Indeed, the additional computation time of NSGAII algorithm and SPEA2 algorithm exceeds respectively 15% and 30% compared to the computation time of our algorithm. As can be seen from Fig. 3, the best improvement rate of computation time can be more than 25% compared to SPEA2 and more than 14% compared to NSGAII. In addition, we notice from Fig. 4 that the improvement rate of computation time while

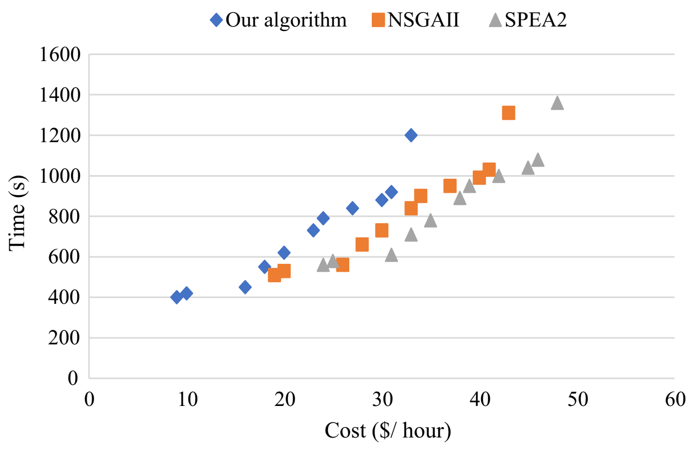

Fig. 5 Cost-time trade-off analysis

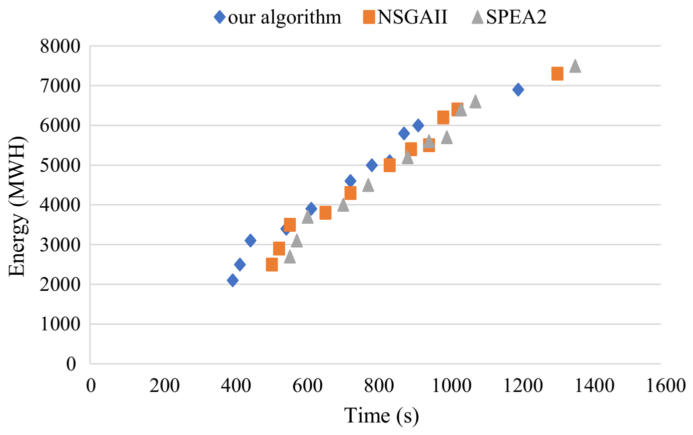

Fig. 6 Time-energy trade-off analysis

increasing the number of resource types reaches 25% compared to SPEA2 and 11% compared to NSGAII.

5.2.2 Evaluation of non-dominated solutions

Figures 5, 6 and 7 show the comparison between the non-dominated solutions of our algorithm and the two well-known multi-objective algorithms namely the Non dominated Sorting Genetic Algorithm II (NSGA II) and the Strength Pareto Evolutionary Algorithm (SPEA2). These figures represent a subset of the identified pareto front solutions related respectively to the cost/time, energy/time and energy/cost objectives. The illustrated results are obtained after applying the three algorithms on a business process composed of 80 tasks.

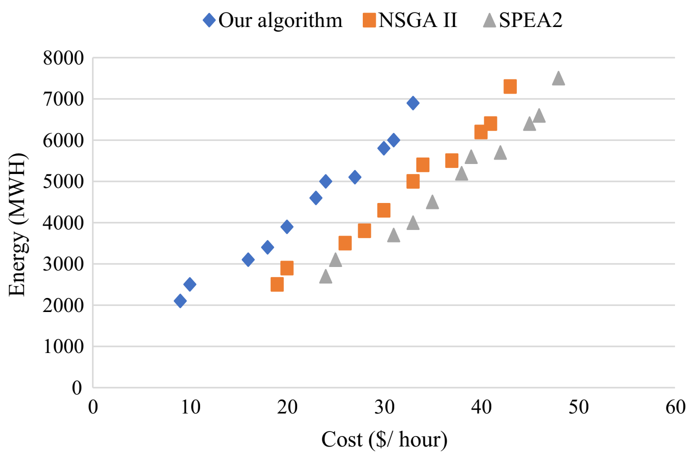

Fig. 7 Cost-Energy trade-off analysis

Table 6 Percentage of business processes meeting the user-defined constraints

	Within time constraints	Out of time constraints	Within budget	Out of budget
Our algorithm	91%	9%	88%	12%
NSGAII	86%	12%	82%	18%
SPEA2	80%	20%	77%	23%

As illustrated in these figures, the pareto front solutions identified after applying our algorithm have better results i.e. minimum values of business process execution's cost, time and energy consumption.

5.2.3 Evaluation of time and budget constraints

To evaluate our algorithm in terms of meeting the user-defined constraints, we show in Table 6 the percentage of business processes which meet their budget and time when executed using the three algorithms. Here, we can see that our algorithm not only achieves less makespan in in terms of low cost as compared with SPEA2 and NSGAII algorithms, but it also satisfies the user-defined constraints. When scheduled using our algorithm, 91% of business processes were executed within the time constraints and 88% met the specified budget. As can be seen from Table 6, the improvement rate in terms of respecting constraints reaches about 5.5% compared to NSGAII algorithm and 11% compared to SPEA2 algorithm. This shows that our proposed algorithm outperforms the two competitive algorithms in terms of the studied metrics.

6 Related work

Many research studies have dealt with the problem of resource allocation for business processes as it is a challenging and an up-to-date issue [17, 37–41]. However, only few works have addressed this problem in an environment based on the collaboration between Cloud and Fog computing.

In [12], the authors proposed an algorithm aiming to optimize the financial cost of Cloud and Fog resources while taking into account the network condition. In addition, a task reassignment strategy which refines the output of the proposed algorithm is presented to satisfy the deadline of the business process.

The approach proposed in [11] introduced a dynamic scheduling heuristic for multiple real-time IoT processes. This heuristic attempts to schedule the tasks which need high computation and low communication requirements in the Cloud. Yet, the communication intensive tasks which need low computational demands are scheduled in the Fog. During scheduling, the possible gaps of Cloud and Fog resources are used to reduce the financial cost and satisfy the user-defined deadline.

The main objective of the scheduling algorithm proposed by Xu et al. [10] is to seek a tradeoff between execution time and resource cost of a business process in a Cloud-Fog environment. To do so, they used PSO algorithm to solve the mapping solution between tasks and resources. However, the authors do not take into account any time constraint to limit the execution duration of the process.

Ding et al. [42] introduced a strategy for scheduling multiple business processes using Fog and Cloud based environment. The goal of this strategy is to reduce the financial cost of resources taking into account the deadline constraint of each process. To do so, the authors used PSO and Min-Min algorithms.

The higher the diversity of Cloud resources, attention must be paid to optimization problems as multi-objective ones. Different researches have been carried out to respond to this trend [43–47].

These researches have taken into consideration the optimization of many attributes that can be classified into functional and non-functional ones. Functional attributes deal with the specific function of the system (i.e., what it is supposed to accomplish), mainly including inputs, behavior and outputs. In addition to functional attributes, it is important to cope with the optimization of non-functional ones (i.e.,how it accomplishes a given functionality) such as cost, reliability, and performance.

In [43], the authors designed a new systematic method for tasks placement in the Cloud that considers both task completion time and security requirements. This work proposed a heuristic algorithm that reduces the overall security threat, risk, while maintaining a reasonable completion time in a heterogeneous Cloud environment. The objective function of this work deals with non-functional properties as computation time and data transmission cost, while guaranteeing data security.

Fard et al. [44] proposed a generic multi-objective optimization framework to evaluate the list scheduling heuristic in distributed computing infrastructures. This approach uses Pareto optimality to generate the best scheduling strategy over four scheduling objectives (i.e. financial cost, execution time, energy consumption and reliability). User-specified constraints on objectives are used to approximate the optimal solution by maximizing and minimizing the distance to users' constraints for dominant solutions. The authors consider non-functional properties such as availability and performance.

Rehman et al. [45] proposed a new Multi-Objective Genetic Algorithm for the scheduling of scientific workflows in Cloud environments. This approach aimed at minimizing non-functional properties such as the energy consumption and the makespan under deadline and budget constraints.

Recently, to solve multi-objective workflow scheduling problems in the Cloud, some hybrid algorithms have been proposed. Anwar and Deng [46] proposed a hybrid metaheuristic based algorithm to optimize multi-objective scheduling of scientific workflows in a Cloud computing environment. This work aimed to to minimize non-functional properties such as makespan, execution cost, and inefficient utilization of the VMs by predicting the earliest finish time (PEFT) and symbiotic organisms search (SOS) algorithms.

In [48], the authors presented a comprehensive analysis of the most recent known autonomous and elastic resource management approaches in the Cloud. They examined the existing elastic and autonomous resource management strategies in terms of their goals, functions, and procedures. Furthermore, they provided a taxonomy, with an emphasis on elastic and autonomous techniques. The paper also briefly discussed the open research difficulties and resource management future trajectory in Cloud computing.

The authors in [13] investigated two semi-greedy based algorithms to solve the IoT tasks scheduling problem in fog environment. The proposed formulation aimed to minimize the total consumed energy of fog nodes, under task execution delay constraints. These methods were shown to achieve good results in terms of total deadline violation time. However, the proposed model did not consider the resource failure problem in Fog nodes. The resource allocation at the Fog and the Cloud was formulated in [14] as a multi-objective optimization problem aiming at providing a tradeoff between energy consumption and task execution delay. The first step takes into consideration the different requirements of edge tasks by allocating low latency tasks to the Cloud. The second step reduced the number of unused time slots between the two scheduled tasks.

The concept of Volunteer Computing System (VCS) was studied by Hoseiny et al. [49]. Fog and Cloud node voluntarily share their extra available resources, where a broker manages request processing, resource monitoring and task scheduling. The authors proposed two heuristics where the objective is to minimize computation, communication, and delay violation costs. The energy consumption is not considered.

In [50], the authors proposed two schedulers to decide where applications should be executed on Fog or Cloud nodes. Their solution used integer linear programming to handle distinct jobs scheduling in cloud or fog resources considering different resources such as processing capacity, memory capacity, and available storage capacity. They used random and round robin methods in their research to provide numerical results.

In [51], the authors tackled the potential of Fog for scheduling workflows with hard time constraints. Furthermore, they offered a new approach called Multi-objective Workflow Offloading (MOWO), a new Pareto-based technique for tasks offloading in Fog. MOWO takes into account three optimization goals: reliability, response time, and financial cost. The proposed scheduler was tested on many real-world applications with severe latency constraints (i.e., biomedical, meteorological, and astronomical workflows). This work did not consider the objectives related to Cloud provider, such as energy usage and cost.

In [47], the authors proposed using multi-objective optimization approach to generate Pareto optimal solutions for scientific workflow scheduling in IaaS Clouds. They proposed a multi-objective hybrid particle swarm optimization (HPSO) algorithm based upon non-dominance sorting procedure to optimize non-functional properties such as cost and makespan depending on the user's preference.

Though there has been some research on resource allocation for business processes in a Cloud-Fog environment, to the best of our knowledge, no previous work has considered the duration constraints of tasks. Furthermore, we notice that the existing multi-objective algorithms are only suitable for the Cloud and/or grid model. Very few works have been done done to solve a multi-objective process scheduling problem in a Fog environment without paying attention to Cloud resources and temporal constraints of business processes. Unlike the work presented above, this paper proposes a multiple objectives particle swarm optimization algorithm based on a non-dominance sort to handle the scheduling problem of business processes with many conflicting objective functions.

7 Conclusion and future work

In this paper, we proposed a new algorithm for scheduling time-aware business processes in a Cloud-Fog environment. Our algorithm simultaneously optimizes three conflicting objectives namely, the makespan, cost and energy under deadline and budget constraints. It is based on the meta-heuristic PSO and it produces a set of non-dominated solutions. In addition, it uses the DVFS technique to reduce energy consumption. The efficiency of our algorithm is demonstrated through comparing it with two competitive algorithms.

In this work, we have considered on demand instances offered by the Amazon Cloud. However, it is interesting to extend our approach to take into account other pricing strategies such as reserved and spot instances. Furthermore, we intend in the future to extend our approach with the notion of configurable business processes.

Acknowledgements This research was funded by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R125), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Data availability Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this paper.

References

1. Wakrime AA (2017) Satisfiability-based privacy-aware cloud computing. Computer J 60(12):1760–1769
2. Abdulkareem N, Zeebaree S, Sadeeq MM, Ahmed D, Sami A, Zebari R (2021) Iot and cloud computing issues, challenges and opportunities: A review. Qubahan Acad J 1:1–7. https://doi.org/10.48161/qaj.v1n2a36
3. Fakhfakh F, Kallel S, Cheikhrouhou S (2021) Formal verification of cloud and fog systems: a review and research challenges. J Univers Comput Sci 27(4):341–363
4. Chaabane M, Bouassida Rodriguez I, Colomo Palacios R, Gaaloul W, Jmaiel M (2019) A modeling approach for systems-of-systems by adapting ISO/IEC/IEEE 42010 standard evaluated by goal-question-metric. Sci Comput Progr 184:871
5. Wang X, Li J, Yang M, Chen Y, Xu X (2018) An empirical study on the factors influencing mobile library usage in iot era. Libr Hi Tech 36(4):605–621
6. Matrouk K, Al-atoun K (2021) Scheduling algorithms in fog computing: a survey. Int J Netw Distrib Comput 9:59. https://doi.org/10.2991/ijndc.k.210111.001
7. Hamdi M, Hamed AB, Yuan D, Zaied M (2021) Energy-efficient joint task assignment and power control in energy harvesting d2d offloading communications. IEEE Internet Things J 1:81
8. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things, In: Proceedings of the first edition of the workshop on Mobile Cloud Computing, ACM, pp 13–16
9. Lin Y, Shen H (2015) Leveraging fog to extend cloud gaming for thin-client mmog with high quality of experience. In: Proceedings of the 35th International Conference on Distributed Computing Systems, IEEE, pp 734–735
10. Xu R, Wang Y, Cheng Y, Zhu Y, Xie Y, Sani AS, Yuan D (2018) Improved particle swarm optimization based workflow scheduling in cloud-fog environment. In: Proceedings of the International Business Process Management Workshops, Springer, pp 337–347
11. Stavrinides GL, Karatza HD (2018) A hybrid approach to scheduling real-time iot workflows in fog and cloud environments. Int J Multim Tools Appl 85:1–17
12. Pham X, Nguyen MD, Tri NDT, Ngo QT, Huh E (2017) A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. Int J Distrib Sensor Netw 13(11):59
13. Azizi S et al (2022) Deadline-aware and energy-efficient iot task scheduling in fog computing systems: a semi-greedy approach. J Netw Computer Appl 201:103333
14. Ijaz S, Munir EU, Ahmad SG, Rafique MM, Rana OF (2021) Energy-makespan optimization of workflow scheduling in fog-cloud computing. Computing 103(9):2033–2059
15. Fakhfakh F, Hadj Kacem H, Hadj Kacem A (2017) Dealing with structural changes on provisioning resources for deadline-constrained workflow. J Supercomput 73(7):2896–2918
16. Ben Halima R, Kallel S, Gaaloul W, Jmaiel M (2018) Scheduling business process activities for time-aware cloud resource allocation. In: Proceedings of the International Conference on the Move to Meaningful Internet Systems, Vol. 11229 of LNCS, Springer, pp 445–462
17. Ben Halima R, Kallel S, Gaaloul W, Jmaiel M (2017) Optimal cost for time-aware cloud resource allocation in business process, In: Proceedings of the IEEE International Conference on Services Computing, IEEE Computer Society, pp 314–321
18. Fakhfakh F, Neji A, Cheikhrouhou S, Kallel S (2019) Optimizing the performance of timed-constrained business processes in cloud-fog environment, In: Proceedings of the International Workshops DETECT, DSSGA, TRIDENT held in Conjuction with the International Conference on New Trends in Model and Data Engineering - MEDI, Vol. 1085, Springer, pp 78–90
19. Conti S, Faraci G, Nicolosi R, Rizzo SA, Schembra G (2017) Battery management in a green fog-computing node: a reinforcement-learning approach. IEEE Access 5:21126–21138
20. Energy-aware load balancing in fog cloud computing, Materials Today: Proceedings (2020)

21. Razaque A, Jararweh Y, Alotaibi B, Alotaibi M, Hariri S, Almi'ani M (2021) Energy-efficient and secure mobile fog-based cloud for the internet of things. Fut Gener Computer Syst 127:61
22. Mezmaz M, Melab N, Kessaci Y, Lee YC, Talbi E-G, Zomaya AY, Tuyttens D (2011) A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. J Parallel Distrib Comput 71(11):1497–1508
23. Hajjej F, Hamdi M, Ejbali R, Zaied M (2019) A new optimal deployment model of internet of things based on wireless sensor networks, In: 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC), pp 2092–2097
24. Deb K (2001) Multi-objective optimization using evolutionary algorithms, vol 16. Wiley, New Jersey
25. Gómez J, Gil C, Baños R, Márquez AL, Montoya FG, Montoya M (2013) A pareto-based multi-objective evolutionary algorithm for automatic rule generation in network intrusion detection systems. Soft Comput 17(2):255–263
26. Kennedy J, Eberhart R (1995) Particle Swarm Optimization, In: Proceedings of the IEEE International Conference on Neural Networks, Vol. 4, pp 1942–1948
27. Kennedy J, Eberhart R (1995) Particle swarm optimization, In: Proceedings of the International Conference on Neural Networks, Vol. 4, IEEE, pp 1942–1948
28. Zhou Z, Chang J, Hu Z, Yu J, Li F (2018) A modified pso algorithm for task scheduling optimization in cloud computing. Concurr Comput: Pract Exper 30(24):e4970
29. Li Z, Ge J, Yang H, Huang L, Hu H, Hu H, Luo B (2016) A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. Fut Gener Computer Syst 65:140–152
30. Zhan S, Huo H (2012) Improved pso-based task scheduling algorithm in cloud computing. J Inf Comput Sci 9(13):3821–3829
31. Morales AK, Quezada CV (1998) A universal eclectic genetic algorithm for constrained optimization, In: Proceedings of the 6th European congress on intelligent techniques and soft computing, Vol. 1, pp 518–522
32. Rekik M, Boukadi K, Ben-Abdallah H (2015) Specifying business process outsourcing requirements, In: Proceedings of the 10th International Joint Conference on Software Technologies, Springer, pp 175–190
33. Yassa S, Chelouah R, Kadima H, Granado B (2013) Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. Scientif World J 2013:61
34. Lee YC, Zomaya AY (2010) Energy conscious scheduling for distributed computing systems under different operating conditions. IEEE Trans Parallel Distrib Syst 22(8):1374–1381
35. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evolut Comput 6(2):182–197
36. Kim M, Hiroyasu T, Miki M, Watanabe S (2004) SPEA2+: Improving the performance of the strength Pareto evolutionary algorithm 2, In: Proceedings of the International Conference on Parallel Problem Solving from Nature, Springer, pp 742–751
37. Xie Y, Chen S, Ni Q, Wu H (2019) Integration of resource allocation and task assignment for optimizing the cost and maximum throughput of business processes. J Intell Manuf 30(3):1351–1369
38. Ihde S, Pufahl L, Goel A, Weske M (2019) Towards dynamic resource management in business processes, In: Proceedings of the 11th Central European Workshop on Services and their Composition, pp 17–23
39. Xu X, Dou W, Zhang X, Chen J (2015) Enreal: an energy-aware resource allocation method for scientific workflow executions in cloud environment. IEEE Trans Cloud Comput 4(2):166–179
40. Fakhfakh F, Hadj Kacem H, Hadj Kacem A (2020) Ensuring the correctness of adaptive business processes: a systematic literature review. Int J Comput Appl Technol 62(3):189–199
41. Lai C, Zhong H, Chiu P, Pu Y (2021) Development and evaluation of a cloud bookcase system for mobile library. Libr Hi Tech 39(2):380–395
42. Ding R, Li X, Liu X, Xu J (2018) A cost-effective time-constrained multi-workflow scheduling strategy in fog computing, In: Proceedings of the International Conference on Service-Oriented Computing Workshops, Vol. 11434, Springer, pp 194–207
43. Abazari F, Analoui M, Takabi H, Fu S (2019) Mows: multi-objective workflow scheduling in cloud computing based on heuristic algorithm, Simul Model Practice Theory 93:119–132
44. Fard HM, Prodan R, Fahringer T (2014) Multi-objective list scheduling of workflow applications in distributed computing infrastructures. J Parallel Distrib Comput 74(3):2152–2165

45. Rehman A, Hussain SS, ur Rehman Z, Zia S, Shamshirband S (2019) Multi-objective approach of energy efficient workflow scheduling in cloud environments. Concurr Comput: Pract Exper 31(8):e4949
46. Anwar N, Deng H (2018) A hybrid metaheuristic for multi-objective scientific workflow scheduling in a cloud environment. Appl Sci 8(4):538
47. Verma A, Kaushal S (2017) A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. Parallel Comput 62:1–19
48. Saif MAN, Niranjan S, Al-Ariki HDE (2021) Efficient autonomic and elastic resource management techniques in cloud environment: taxonomy and analysis. Wireless Netw 27(4):2829–2866
49. Hoseiny F, Azizi S, Shojafar M, Tafazolli R (2021) Joint qos-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system. ACM Trans Internet Technol 21(4):6451
50. Guevara J, Fonseca N (2021) Task scheduling in cloud-fog computing systems. Peer-to-Peer Netw Appl 14:962. https://doi.org/10.1007/s12083-020-01051-9
51. De Maio V, Kimovski D (2020) Multi-objective scheduling of extreme data scientific workflows in fog. Fut Gener Computer Syst 106:171–184

Authors and Affiliations

Fairouz Fakhfakh[1,2] · Saoussen Cheikhrouhou[1] · Bouthaina Dammak[3] · Monia Hamdi[4] · Mouna Rekik[5]

Fairouz Fakhfakh
Fairouz.Fakhfakh@redcad.tn

Saoussen Cheikhrouhou
Saoussen.Cheikhrouhou@redcad.tn

1 ReDCAD, ENIS, University of Sfax, Sfax, Tunisia

2 ISMAIK, University of Kairaoun, Kairaoun, Tunisia

3 Department of Computer Science and Information Technology, Applied college, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia

4 Information Technology Department, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia

5 MIRACL, ISIMS, University of Sfax, Sfax, Tunisia