Computational Intelligence Based Heuristic Approach for Maximizing Energy Efficiency in Internet of Things

Amandeep Verma, Sakshi Kaushal and Arun Kumar Sangaiah

Abstract Internet of Things (IoT) is a network of physical things or objects fixed with electronics, software, sensors and network connectivity. It is a new revolution of the Internet that is hastily gathering impetus driven by the advancements in sensor networks, mobile devices, networking, and wireless and cloud technologies. It permits these things or objects to collect and transact and analyze data for performing specific tasks in digital world with limited storage and processing capacities. Due to massive adoption of cloud computing having virtually unlimited storage and processing capabilities, it can be merged with IoT to be an important component of Future Internet. This chapter explores IoT and cloud computing as well as their symbiosis based on the common environment of distributed processing. For IoT devices to be operational for longer time, the development of energy efficient schemes for sustainable computing environment is a challenging issue. It is possible to lease on-demand computing resources through cloud in an optimized manner from energy point of view. Further, Computational Intelligence can help to save device resources and energy by shifting the computational tasks from device to cloud. To make the devices energy efficient, this chapter also presents a dominance sort based optimized heuristic to offload and process local computations on cloud. The proposed approach uses the multi-objective swarm intelligence technique, i.e., Multi-Objective Particle Swarm Optimization (MOPSO) to generate Pareto optimal solutions for task offloading. Our method first determined which of the tasks can be run locally over the mobile cores and which are to be offloaded to the cloud. This will result into lower cost and high value to end user of services. The overall outcome can be helpful to bolster the performance of IoT devices. Higher operational efficiency in

A. Verma (✉) · S. Kaushal
UIET, Panjab University, Chandigarh, India
e-mail: amandeepverma@pu.ac.in

S. Kaushal
e-mail: sakshi@pu.ac.in

A.K. Sangaiah
School of Computing Science and Engineering, VIT University,
Vellore, Tamil Nadu, India
e-mail: arunkumarsangaiah@gmail.com

A.K. Sangaiah et al. (eds.), *Intelligent Decision Support Systems for Sustainable Computing*, Studies in Computational Intelligence 705,
DOI 10.1007/978-3-319-53153-3_4

system will help and support to make IoT sustainable in long run. These improvements will pave the way for achieving a new level in IoT industry and establishing new standard for benchmarking.

Keywords Computational intelligence · Sustainable computing · IoT · Cloud · Non-dominance · Pareto optimal solution

1 Introduction

Cloud computing is the latest emerging paradigm of distributed computing which uses the concept of hardware and software virtualization to provide a dynamically scalable services. Based upon the demand, these services can be accessed over Internet. In the past few years, Distributed and Parallel Computing as well as Service Oriented Computing attract the interests of researchers [1]. As compared with the other traditional computing paradigms like cluster, Grid, and peer-to-peer (p2p), the Cloud computing adopts a market-oriented business model where users are charged for consuming Cloud services such as computing, storage, and network services like conventional utilities in everyday life (e.g. water, electricity, gas, and telephony) [2]. Cloud computing delivers three defined models as *Software as a Service* (*SaaS*), where the user uses the various applications but has no control over the hosting environment. The examples of SaaS include Google Apps and Salesforce.com [3]. *Platform as a Service* (*PaaS*) offers a full or partial application development environment that users can access and utilized online collectively or individually. It facilitates the deployment of applications without the cost and complexity of buying and managing the underlying hardware and software and provisioning hosting capabilities. In this model, the platform is typically an application framework. AWS and Google App Engine are PaaS cloud providers [4]. In infrastructure as a service (*IaaS*), the service provider provides the wide variety of resources of different processing power and storage capabilities. The user can use these resources to deploy its own applications. The user no longer needs to maintain the hardware. Amazon EC2, Globus, Nimbus, and Eucalyptus are *IaaS* providers [5]. Scalability and flexibility are two main advantages of cloud where the user can access and release the resources as per their need.

On the other hand, Internet of Things (IoT) is a network of physical things or objects fixed with electronics, software, sensors and network connectivity. It is based upon ubiquitous and pervasive computing [6]. IoT consists of real world small things with limited processing and storage capacities. But, the cloud virtually provides unlimited storage and processing capabilities. Thus, by integrating these two complementary techniques, the mutual advantages have been identified in the literature and this new paradigm is known as CloudIoT [7]. In general, to handle the issues of limited processing and storage capabilities, IoT can use unlimited resources of Cloud. Similarly, Cloud can extend its scope from virtual to real world things with the help of IoT. Cloud acts as an intermediate between the real things

and virtual applications. It hides the complex functionality details to implement the IoT [8]. The various applications of IoT includes Healthcare services [9], Smart cities and communities [10], Smart home [11], Video surveillance [12], and Energy efficient smart grid [13] etc. All of these applications need massive storage and computational resources. Combining IOT with Cloud can solve the problem of processing and storage. In the following we list the few advantages obtained using CloudIoT paradigm:

(i) Storage: IoT applications generate a large volume of structured and semi-structured data. It requires collecting, processing, sharing and searching of large volume of data. This problem can be solved by accessing the unlimited, cost efficient and on-demand storage services of Cloud [14].
(ii) Computational Resources: IoT devices cannot perform complex on-site data processing due to their limited processing and limited battery. So, major processing unit of an application is transmitted to nodes that are more powerful in terms of processing and storage. As, cloud provides virtually unlimited processing capabilities, this represents another important CloudIoT driver. The major processing part of an application is offloading to the cloud for energy saving of IoT devices [7].

Thus, these several motivations lead to the integration of Cloud and IoT. But, at the same time, it imposes several challenges for each application. Main challenges include heterogeneity of resources, security, performance, reliability and power and energy efficiency, etc. [8].

For IoT devices to be operational for longer time, the development of energy efficient schemes for sustainable computing environment is a challenging issue. Energy efficient task offloading is currently getting interest of the research community [15]. Computational Intelligence techniques can help to save device resources and energy by shifting the computational tasks from device to cloud. Computational Intelligence techniques generate number of Pareto optimal solutions depending upon the user requirements.

This chapter first explores the work done in the area of energy efficient task offloading techniques for IoT enabled mobile devices. From the study of work done, we have found that none of the existing techniques used Computational Intelligence techniques to give optimal energy saving results to the user. So, to make the devices energy efficient, in this chapter, we have also proposed a novel technique to offload the task from IoT enabled mobile devices to cloud. The proposed approach uses the multi-objective computational intelligence to generate Pareto optimal solutions for task offloading. Our method first determined which of the tasks can be run locally over the mobile cores and which are to be offloaded to the cloud. Then, results of this assignment is fed into the initial population of multi-objective swarm intelligence technique, i.e., Multi-Objective Particle Swarm Optimization (MOPSO) to schedule the task either over mobile cores or offloaded to cloud such that the precedence requirements among the different tasks along with time constraints are met and energy consumption of IOT mobile devices is

minimized. The simulation analysis validates that the solutions obtained with proposed heuristic deliver better convergence and uniform spacing among the solutions as compared to others.

The remaining chapter is organized as follow: Sect. 2 presents the related work done on energy efficient task offloading techniques. The problem description is presented in Sect. 3. Section 4 described the multi-objective optimization approach. Section 5 explains the proposed modified multi-objective PSO. Section 6 discusses the simulation strategy and result analysis. Finally, Sect. 7 concludes the chapter.

2 Related Works

The various heuristics in the literature have been proposed for task scheduling and task offloading problems in mobile cloud environment. Broadly, these are of two types: (I) minimizing the makespan of an application [16–18] and (ii) minimizing the battery consumption of mobile devices [15, 19, 20]. With the objective of minimizing the makespan, a list based heuristic, HEFT[16] was proposed. Firstly, it assigned priority to all tasks and then mapped the highest priority task to a machine that gave the earlier finish time of a task at each step and thus minimized the overall completion time of an application. Another heuristic, *named*, Push-Pull algorithm has also been proposed. This algorithm initially used a random schedule and then deterministic guided search method is applied to iteratively improve the current solution [17]. For maximizing the throughput a genetic algorithm was proposed [18] that partition the application task over a mobile device and the cloud in an optimized manner. An incremental greedy strategy to offload and parallel execution of perceptual applications [21] has also been proposed to reduce the finish time of applications. In recent years, the main focus of the researchers is on energy aware scheduling mobile devices. Rong and Pedram [22] used the positive slack time between tasks for minimizing the energy consumption of a computer system. Li et al. [23] presented an optimized maximum-flow/minimum-cut task partitioning algorithm to offload the tasks from mobile device to cloud for minimizing energy consumption. Kumar and Lu [24] proposed a strategy based upon computation-to communication ratio for making offloading decision to minimize the energy consumption.

To find the trade-off solutions between completion time and energy for parallel tasks, Lee and Zomaya [25] proposed two energy-conscious scheduling (ECS and ECS + idle) for heterogeneous computing systems. An integer liner programming based optimization technique [26] for adaptive computation offloading is addressed considering the available memory, CPU and energy consumption as the main criteria for offloading. Wu et al. [27] presented an offloading decision model using network unavailability to decide whether to offload a task for remote execution or not. Similarly, another task offloading technique, CRoSS algorithm [28] was also presented using the link failure rate and the bidirectional transmission rate as main factors for offloading. Along with these factors, other important computing factor is

clock frequency. Using Dynamic Voltage Frequency Scale (DVFS), the mobile device energy can be further optimized [29] as the CPU clock frequency is approximately linearly proportional to the voltage supply. Similarly, a mobile device can be connected to more than one wireless networks and thus can offload the data to different networks. To address multisite offloading, a graph partitioning approach is proposed to find solution to the partitioning problem [30]. Lin et al. proposed the task scheduling and task migration algorithm from mobile device to cloud based on DVFS for mobile cloud computing [31]. The results showed a significant reduction in energy under the application completion constraints. Energy efficient computational offloading framework (EECOF) [32] has been presented to leverage minimal application processing migration to cloud and thus reducing the total energy consumption cost. Based on contextual network conditions [33], an energy model was presented whether to offload a task or to run it locally. Similarly, few fuzzy and artificial intelligence based decision support systems [34–36] have also been developed to offload the tasks.

From the review of literature, it has been found that most of the existing studies try to minimizing the makespan or energy consumed while scheduling the tasks in mobile cloud environment. None of the existing techniques used multi-objective computational intelligence techniques like MOPSO [37], NSGA-II [38], and FDPSO [39] etc. that give set of near optimal solutions. Hence, this chapter presented multi-objective optimization technique that generates a set of near optimal solutions for mobile cloud applications. We proposed the Modified Multi-Objective Particle Swarm Optimization (MMOPSO) algorithm using the concept of non-dominance to offload the tasks from mobile device to the cloud so as to minimize the energy consumption of created schedule plan.

3 System Model and Assumptions

3.1 Application Model and Mobile Cloud Model

A user application is modelled by a Directed Acyclic Graph (DAG), defined by a tuple $G\ (T, E)$, where T is the set of n tasks $\{t_1, t_2, \ldots, t_n\}$, and E is a set of e edges, represent the dependencies. Each $t_i\ \varepsilon\ T$, represents a task in the application and each edge $(t_i \ldots t_j)\ \varepsilon\ E$ represents a precedence constraint, such that the execution of $t_j\ \varepsilon\ T$ cannot be started before $t_i\ \varepsilon\ T$ finishes its execution [40]. If $(t_i, t_j)\ \varepsilon\ T$, then t_i is the parent of t_j, and t_j is the child of t_i. A task with no parent is known as an *entry* task and a task with no children is known as *exit* task. The task size (z_i) is expressed in *Million of Instructions* (*MI*).

Our mobile cloud model consists of a mobile device having m, computational cores, $R = \{r_1, r_2, \ldots, r_m\}$ at different processing power and a cloud resource. The processing power of a core (mobile core or cloud core), is expressed as *Million of Instruction per Second* (*MIPS*) *and is denoted by* PP_{r_p}. Each core is Dynamic Voltage Scaling (DVS) enabled; in other words, it can operate with different

Voltage Scaling Levels (VSLs) i.e., at different clock frequencies. This mobile device has also access to the computing resources on the cloud. Each task can be executed on different cores or can be offloaded to cloud for its execution. The execution time, $ET(_{i,p})$, of a task t_i on a core (either mobile core or cloud core), is calculated by the following equation:

$$ET_{(i,p)} = \frac{Z_i}{PP_{r_p}} \tag{1}$$

We use $ET(_{i,c})$ to denote the execution time of task t_i on cloud c. Time for sending the task t_i to cloud is given by

$$T_s^i = \frac{data_i}{BW_s} \tag{2}$$

where $data_i$ is the task data and BW_s is the available bandwidth of sending channel. Similarly, time for receiving output of task t_i from cloud is given by

$$T_r^i = \frac{data_i}{BW_r} \tag{3}$$

where $data_i$ is the task data and BW_r is the available bandwidth of receiving channel.

Let $EST\ (t_i,\ r_p)$ and $EFT\ (t_i,\ r_p)$ denote the earliest Earliest Start Time and the Earliest Finish Time of a task t_i on a local core r_p, *respectively*. For the entry task, we have:

$$EST\left(t_{entry}, r_p\right) = avail\left(r_p\right) \tag{4}$$

For the other tasks in DAG, we computer EST and EFT recursively as follows:

$$EST\left(t_i, r_p\right) = max\begin{cases} avail\left(r_p\right) \\ \underset{t_j \varepsilon pred(t_i)}{max}\left\{AFT\left(t_j\right) + ct_{ij}\right\} \end{cases} \tag{5}$$

$$EFT\left(t_i, r_p\right) = ET_{(i,p)} + EST\left(t_i, r_p\right) \tag{6}$$

where pred (t_i) is the set of parent tasks of task t_i, and *avail* (r_p) is the time when the core r_p is ready for task execution. The Estimated Remote Execution Time of a task t_i on a cloud is given by:

$$ERT(t_i, c) = ET_{(i,c)} + T_s^i + T_r^i \tag{7}$$

Similarly, *AST* $(t_i r_p)$ and *AFT* $(t_i r_p)$ denotes the Actual Start Time and Actual Finish Time of task t_i on local core or on cloud, *respectively.* The makespan is equal to the maximum of actual finish time of the exit tasks t_{exit} and is defined by

$$M = max\{AFT(t_{exit})\} \tag{8}$$

The makespan is also referred to as the running time for the entire application DAG. The energy model used in this study is derived from the capacitive power (P_c) of Complementary Metal-Oxide Semiconductor (CMOS)-based logic circuits [41] which is given by:

$$P_c = ACV^2 f \tag{9}$$

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. It's clear from Eq. (9) that the supply voltage is the dominant factor; hence, low supply voltage means lower power consumption.

The energy consumed by executing entire application tasks over available local core is defined as [41]

$$E_l = ACV^2 f \cdot ET_{(i,p)} = \alpha V_i^2 ET_{(i,p)} \tag{10}$$

where V_i is the supply voltage of the core on which task n_i is executed, and $ET_{(i,p)}$ is the execution time of task n_i on the scheduled core r_p.

If task n_i is offloaded to the cloud, the energy consumption of mobile device for offloading the task is given by:

$$E_c = ACV^2 f \cdot ET_{(i,c)} = \alpha V_i^2 ET_{(i,c)} \tag{11}$$

where V_i is the supply voltage of the sending channel and $ET_{(i,c)}$ is the execution time of task n_i on the cloud c. Therefore, the total energy consumed, i.e., E_{total} for executing the whole application is given by

$$E_{total} = \sum_{i=1}^{n} E_i \tag{12}$$

where $E_i = E_l$ if the task t_i is executed locally and is equal to E_c if the task t_i is offloaded to the cloud.

4 Task Scheduling Based on Multi-objective Particle Swarm Optimization

The first part of this section introduces the concept of Multi-Objective Optimization and the second part gives an overview of Particle Swarm Optimization (PSO).

4.1 Multi-objective Optimization

A Multi-objective Optimization Problem (MOP) [42] with m decision variables and n objectives can be formally defined as:

$$Min(y = f(x) = [f_1(x), \ldots, f_n(x)])$$

where $x = (x_1, \ldots, x_m) \in X$ is an m-dimensional decision vector, X is the search space, $y = (y_1, \ldots, y_n) \in Y$ is the objective vector and Y the objective-space.

In general MOP, there is no single optimal solution with regards to all objectives. In such problems, the desired solution is considered to be the set of potential solutions which are optimal for one or more objectives. This set is known as the Pareto optimal set. Some of the Pareto concepts used in MOP are as follows:

(i) *Pareto dominance*. For two decision vectors x_1 and x_2, dominance (denoted by $\prec$) is defined as follows:

$$x_1 \prec x_2 \Leftarrow \Rightarrow \forall_i f_i(x_1) \leq f_i(x_2) \wedge \exists_j(x_1) < f_i(x_2)$$

The decision vector x_1 is said to dominate x_2 if and only if, x_1 is as better as x_2 for all the objectives and x_1 is strictly superior to x_2 in at least one objective.

(ii) *Pareto optimal set*. The Pareto optimal set P_s is the set of all Pareto optimal decision vectors.

$$P_S = \{x_1 \in X, |\nexists x_2 \in : x2 \prec x1\}$$

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.

(iii) *Pareto optimal front*. The Pareto optimal front P_F is the image of the Pareto optimal set in the objective space.

$$P_F = \{f(x) = (f_1(x), \ldots, f_n(x)) | x \in P_S\}$$

4.2 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a stochastic optimization technique that operates on the principle of the social behavior of swarms of birds or the schools of fish [43]. In this technique, a swarm of individuals, known as the particles, flow

through the swarm space. Each particle represents a candidate solution to the given problem. Each particle is associated with two parameters, *namely,* current position, x_i and current velocity, v_i.

The position of a particle is influenced by the best position visited by it, i.e., its own experience (*pbest*). Along with *pbest*, the second parameter that influences the position is the position of the best particle in its neighborhood, i.e., the experience of neighboring particles (*gbest*). The performance of each particle is measured using a fitness function that varies depending on the optimization problem. During each PSO iteration k, particle i updates its velocity v_i^k and position vector x_i^k as described below [43]:

(a) *Updating Velocity Vector*

$$v_i^{k+1} = \omega v_i^k + c_1 rand_1 * \left(pbest_i - x_i^k\right) + c_2 rand_2 * \left(gbest - x_i^k\right) \tag{13}$$

where ω: inertia weight; c_1: cognitive coefficient based on particle's own experience; c_2: social coefficient based on the swarms experience; $rand_1, rand_2$: Random variables with between (0,1).

The inertia weight, ω, controls the momentum of the particle. Improvement in performance is obtained by decreasing the value of ω linearly from its maximum value, ω_1, to its minimum value, ω_2 [44]. At iteration k, its value, ω_k is obtained as:

$$\omega_k = (\omega_1 - \omega_2) \frac{\mathbf{max}\_k - k}{\mathbf{max}\_k} + \omega_2 \tag{14}$$

Similarly, if c_1 decreases from its maximum value, $c_{1max,}$ to its minimum value, c_{1min}, then more divergence among the particles in the search space can be achieved, while if c_2 increases from its minimum value, c_{2min}, to its maximum value, c_{2max}, then the particles are much closer to the present *gbest*. The following equations are used to find the values of c_{1i} and c_{2i} at iteration k:

$$c_{1i} = (c_{1min} - c_{1max}) \frac{k}{\mathbf{max}\_k} + c_{1max} \tag{15}$$

$$c_{2i} = (c_{2max} - c_{2min}) \frac{k}{\mathbf{max}\_k} + c_{2min} \tag{16}$$

where *max_k* is the maximum number of iterations and k is the iteration number.

(b) *Updating Position Vector*

$$x_i^{k+1} = x_i^k + v_i^k \tag{17}$$

where x_i^k: position of the particle at kth iteration; $v_{i:}^k$ velocity of the particle at kth iteration.

(c) **Fitness Function**

The fitness function used in proposed MMOPSO is as described in Eq. (18):

$$Fitness = E_{total} \tag{18}$$

The next section describes the proposed algorithm based upon multi-objective PSO.

5 Proposed Work

In order to solve the multi-objective task scheduling problem for mobile cloud environment, we have proposed the Modified Multi-Objective Particle Swarm Optimization (MMOPSO) algorithm based upon non-dominance sorting procedure. The proposed algorithm is consisting of two phases. In the first phase, the initial schedule in created based upon HEFT [16] to minimizes the makespan. Then in the second phase, the schedule created in first phase is fed into the initial population of MMOPSO for minimizing the energy consumption (E). Both of the phases are explained below:

5.1 First Phase: Initial Schedule

For creating the initial schedule, HEFT algorithm is used to schedule tasks over the mobile cores as well as over the cloud cores. For this purpose, first of all, the application tasks are divided into either local task or cloud task. For each task t_i, we defined its minimum completion time over mobiles cores as

$$T_i^{min} = \min_{1 \le p \le m} \{ET_{(i,p)}\} \tag{19}$$

And if $T_i^{min} < ERT(t_i, c)$, then the task t_i will run on mobile cores and is known as *local task*, otherwise it is known *cloud task*.

If a task t_i is cloud task, then its average execution time is given by

$$w_i = ERT(ti, c) \tag{20}$$

Otherwise,

$$w_i = \underset{1 \le p \le m}{avg} \{ET_{(i,p)}\} \tag{21}$$

Each task is assigned a priority using upward rank as defined in HEFT and is given by Eq. (22).

$$rank(t_i) = w_i + \max_{t_j \epsilon succ(t_i)} \{rank(t_j)\} \tag{22}$$

where w_i is the average execution time of the task on the different computing resources; $succ(t_i)$ includes all the children tasks of t_i. After assigning the rank to all tasks, initial schedule is generated using HEFT.

5.1.1 An Example

An example workflow with 10 tasks as shown in Fig. 1a is considered to illustrate the working of the first phase. Figure 1b shows the execution time of these tasks on three different available mobile cores. It has been assumed that $T_s^i = 3$, $T_r^i = 1$ and $ET(t_i,c) = 1$ for each task.

After applying Eq. (18), only task t_2 is identified as *cloud* task and rest will be assigned on the mobile cores. Then the rank of all the tasks is calculated using Eq. (21). The order of execution after sorting tasks in descending order of their rank is: t_1, t_3, t_6, t_2, t_4, t_5, t_7, t_8, t_9, *and* t_{10}. Now the tasks are assigned either on local cores or over cloud using HEFT as shown in the Fig. 2.

5.2 *Second Phase: MMOPSO Algorithm*

The main steps followed in MMOPSO algorithm are described in Fig. 1. The fitness function used is presented by Eq. (18) in Sect. 4.2. MMOPSO algorithm is executed for bi-objective task offloading problem, i.e., minimization of execution time and energy. Therefore the task offloading problem is formulated as (Fig. 3):

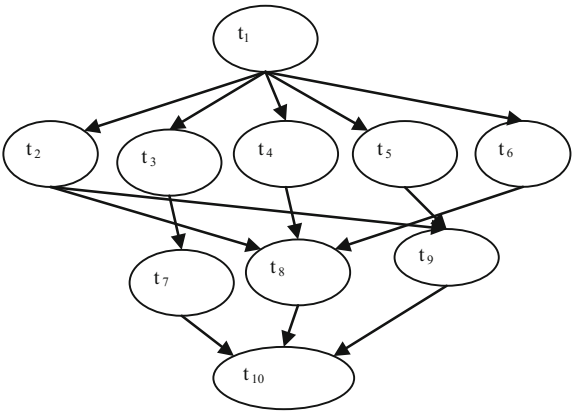
$Minimize\ Time(S) = max\{AFT(t_{exit})\}$
$Minimize\ Energy\ (S) = E_{total}$
$Subject\ to\ Time\ (S) < D$

Where D is the maximum completion time of an application over mobile device. MMOPSO algorithm used the following operators:

(a) *Archive Updating:*

In multi-objective algorithms, the non-dominated particles are stored in elite archive. Particle's dominance is checked against other particles based upon the

Fig. 1 An example



(a) An Example Workflow

Task	Core1	Core2	Core3
t_1	9	7	4
t_2	8	6	5
t_3	6	5	4
t_4	7	5	3
t_5	5	4	2
t_6	7	6	4
t_7	8	5	3
t_8	6	4	2
t_9	5	3	2
t_{10}	7	4	2

b) Execution Time of tasks on different cores

objective functions. The current generation's solutions are combined with the solutions in the archive of previous generations to make *2 N* solutions, where *N* is the size of archive. Then, all of these solutions are sorted in ascending order of their dominance. If more than one solutions show the same dominance value, then diversity perimeter, *I* (.) is calculates for such solutions. The solution showing higher value of *I* (.) is selected. For updating archive, the best *N* solutions are selected from these 2 *N* solutions based upon dominance and perimeter [39].

(b) *Diversity Perimeter*:

The diversity parameter for any solution *y*, *I*(*y*) is given by:

$$I(y) = \sum_{i=1}^{M} \frac{f_i(x) - f_i(z)}{max(f_i) - min(f_i)} \tag{23}$$

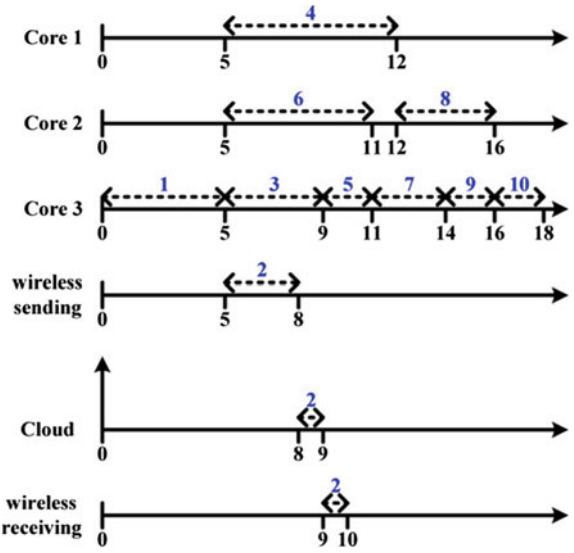


Fig. 2 Initial assignment of first phase

where x and z are adjacent solutions to y, after sorting the solutions in ascending order according to ith objective. The infinite value is assigned to the boundary solutions. Higher the value of $I(y)$, more is sparseness. So, the diversity of the solutions increases with the high values of $I(y)$.

(c) *Updating pbest and gbest:*

The binary tournament operator is used to select *gbest* solution from the current archive. The particle's current position is compared based on dominance sort with the best position from the previous generation for updating *pbest.* If there is no dominate solution, then the current position of particle is selected as current *pbest.*

(d) *Mutation:*

MMOPSO algorithm used the replacement mutation [45] to avoid stucking into local minima and to explore the search space efficiently. For applying the adaptive mutation, mutation probability, $P(Mutation)$ is calculated using the following equation:

$$P(Mutation) = 1 - \frac{k}{max\_k} \tag{24}$$

where k is the current iteration and max_k is the maximum iterations. A random number (*rand*) in range [0, 1] is generated for every particle. If $rand < P$ (*Mutation*), then a task is randomly selected for mutation.

Algorithm : Modified Multi-Objective Particle Swarm Optimization(MMOPSO) algorithm

Input: Application with maximum completion time constraint, D
Output : Non-dominant energy efficient schedule solutions

1. a) Set iteration counter $k = 0$.

   b) Randomly initialize population of N swarm particles.

   c) Insert the schedule created in first phase as one of the swarm particle.

   d) Initialize all particle velocities $V^k_{(i,)}$ to zeros and personal best position $pbest^k_{(i,)}$ is set to $X^k_{(i,)}$.

2. Evaluate the fitness of all swarm particles according to eq. (18).

3. Based on non-dominance and diversity parameter, sort all the particles in ascending order Then, initialize the archive $A^k_{(i,)}$ with it.

4. Set $k = k + 1$.

5. For all the particles, repeat the following:

   a) Initialize the $gbest^k_{(i,)}$ from the archive with binary tournament selection.

   b) Update the velocity of k^{th} particle $V^k_{(i,)}$ according to Eq. (13).

   c) Update particle position $X^k_{(i)}$ according to Eq. (17).

   d) Apply adaptive mutation using Eq. (24)

6. Evaluate the fitness of all swarm particles according to eq. (17).

7. Combine the solutions of current particle positions with the archive solutions to have total of $2N$ particle solutions.

8. Select the best N solutions from the solutions of step 7 on the basis of non-dominance sort and perimeter, acc. to Eq. (23) and update the archive.

9. Update each particles $pbest^k_{(i,)}$ and $gbest^k_{(i,)}$.
10. If ($k <$ max _k) then go to step 3, otherwise output the non- dominant solutions from the archive

Fig. 3 MMOPSO Algorithm

6 Performance Evaluations

In this section, the simulation of the proposed heuristic, MMOPSO is presented. To evaluate the proposed task offloading workflow scheduling algorithm, we used five synthetic workflows based on realistic workflows from diverse scientific applications, which are:

- Montage: Astronomy
- EpiGenomics: Biology
- CyberShake: Earthquake
- LIGO: Gravitational physics
- SIPHT: Biology

The detailed characterization for each workflow including their structure, data and computational requirements can be found in [46]. Figure 4 shows the approximate structure of each workflow.

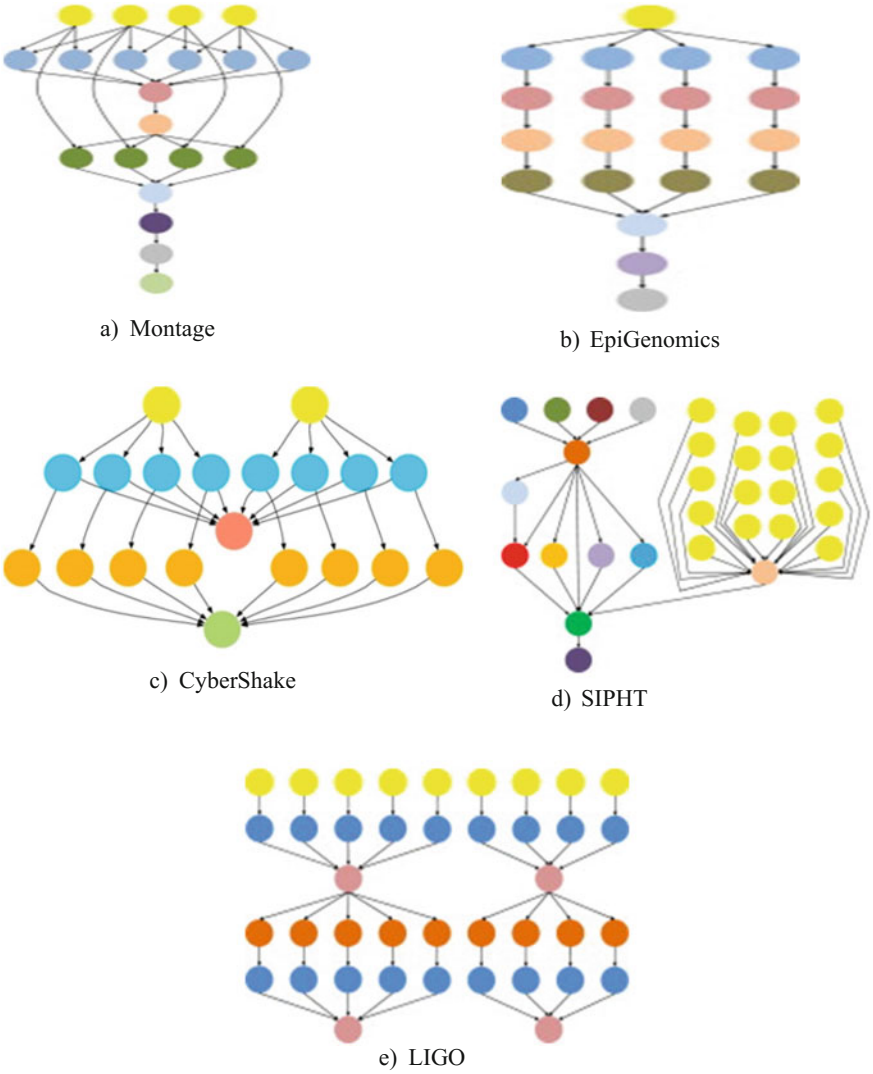


Fig. 4 Structure of various workflows [46]

6.1 Experimental Setup

For simulation, we assume a mobile cloud environment consisting of a mobile device and a cloud service provider. We assume three heterogeneous cores with different processing speed in a mobile device and one core available at the cloud. For simplicity it is assumed that every task take 30 secs to sending data over cloud from the mobile device and take 10 secs to receive the data from the cloud. For this

Table 1 Voltage–relative speed pairs

Level	Pair 1		Pair 2		Pair 3	
	Voltage (v_i)	Relative speed (%)	Voltage (v_i)	Relative speed (%)	Voltage (v_i)	Relative speed (%)
0	1.6	100	2.5	100	2.0	100
1	1.4	85	2.0	75	1.6	80
2	1.2	60	1.5	55	1.4	60
3	1.1	45	1.0	35	1.0	40
4	1.0	30	1.0	35	–	–

study, we have used the CloudSim [47] library. The existing CloudSim simulator allows modelling and simulating cloud environment by dealing only with single workload. It is not suitable for mobile cloud environment. So, the core framework of CloudSim simulator is extended to handle task scheduling problem for MCC. Each core is Dynamic Voltage Scaling (DVS) enabled; i.e., it can work at different Voltage Scaling Levels (VSLs). For each resource, a set V_j of v VSLs is random and uniformly distributed among three different sets of VSLs (Table 1).

The values for maximum completion time, D is generated as:

$$D = 3 * M_{HEFT}$$

where $M_{HEFT} = makespan\ of\ HEFT$

6.2 Performance Metrics

The analysis of the proposed algorithm has been done with existing state-of-art algorithms using the following performance metrics:

(a) Generational Distance (GD): *GD* [38] is a convergence metric and used to access the quality of an algorithm against the true pareto front P^* which is generated by merging solutions of different algorithms. It is calculated using Eq. (25):

$$GD = \frac{(\sum_{i=1}^{|Q|} d_i^2)^{1/2}}{|Q|} \tag{25}$$

where d_i is the Euclidean distance between the solution of Q and the nearest solution of P^*. Q is the front obtained from algorithms for which *GD* metric is calculated.

(b) Spacing: To check the diversity among the solutions, spacing metric [38] is used and is given by Eq. (26):

$$Spacing = \sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}(d_i - \overline{d})^2} \tag{26}$$

where d_i is the distance between the solution and its nearest solution of Q and it is different from Euclidean distance and $\overline{d}$ is the mean value of the distance measures d_i. The small value of both *GD* and Spacing metric is desirable for an evolutionary algorithm.

6.3 Simulation Results

This section presents simulation results and analysis of our proposed Multi-objective MMOPSO algorithm. Now a days, Non-dominated Sort Genetic Algorithm (NSGA-II) [38] and ε-FDPSO [39] are the state-of- art techniques to solve MOP. To measure the effectiveness of proposed MMOPSO algorithm, all these algorithms have been designed and simulated for multi-objective workflow scheduling problem for mobile cloud environment. For implementing the NSGA-II, we used binary tournament selection, one-point crossover and replacing mutation. We have assumed parameters used in ε-FDPSO, and MMOPSO algorithms to be: population size = 20, $c_1 = 2.5 \rightarrow 0.5$ and $c_2 = 0.5 \rightarrow 2.5$, inertia weight $\omega = 0.9 \rightarrow 0.1$ and for NSGA-II population size is 20, crossover rate is 0.8, and mutation rate is 0.5. The performance of scheduling algorithms is evaluated considering the randomly generated workflow applications. For bi-objective task offloading problem, we considered the application completion time and the energy consumed of the created schedule as two conflicting objectives. To obtain the Pareto optimal solutions with ε-FDPSO, MMOPSO, and NSGA-II, algorithms, 100 samples have been captured through simulation.

Figures 5, 6, 7, 8 and 9 shows the bi-objectives non-dominated solutions for Montage, CyberShake, EpiGenomics, LIGO, and SIPHT workflows, *respectively*. The x-axis represents the execution time of created schedule for respective workflow structure, and y-axis represents the energy consumed by created schedule for respective workflow structure.

It has been observed that most of the solutions obtained using MMOPSO algorithm is lying closely to the true front and showing he uniform spacing among the solutions. These results are analyzed using two metrics, i.e., GD and Spacing. Table 2 and Table 3 presents the comparative results for all the three algorithms on the basis of GD and Spacing metrics for Montage, CyberShake, EpiGenomics,

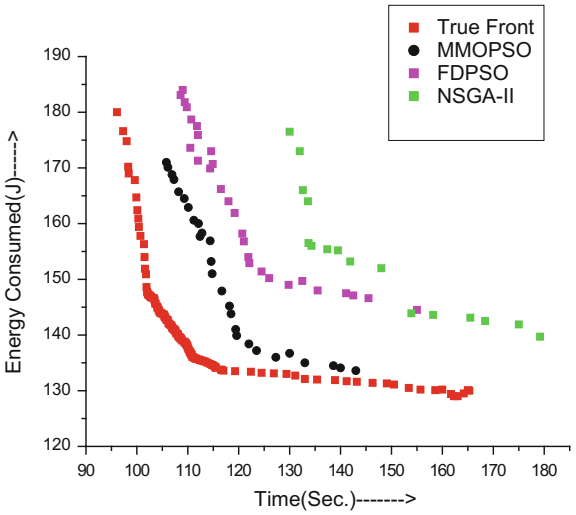


Fig. 5 Bi-objective non-dominated solutions for montage workflow

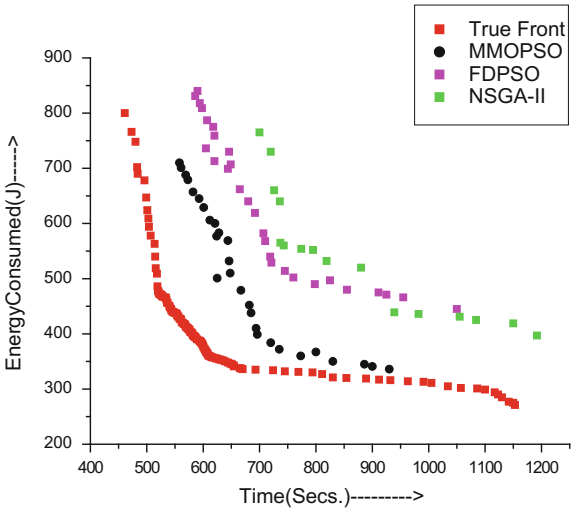


Fig. 6 Bi-objective non-dominated solutions for CyberShake workflow

LIGO, and SIPHT workflows, *respectively*. The results are obtained by taking the average of 10 simulations as described below.

From Table 2, it is clear the performance of the MMOPSO algorithm is better and reaches a solution set that is 84, 83, 55, 55, and 80% closer to true Pareto front as in comparison to the solution set created by FDPSO for Montage, CyberShake, Epigenomics, LIGO and SIPHT workflows, *respectively* as well as 90, 86, 70, 72, and 87% closer to true Pareto front as in comparison to the solution set created by NSGA-II for Montage, CyberShake, Epigenomics, LIGO and SIPHT workflows, *respectively.*

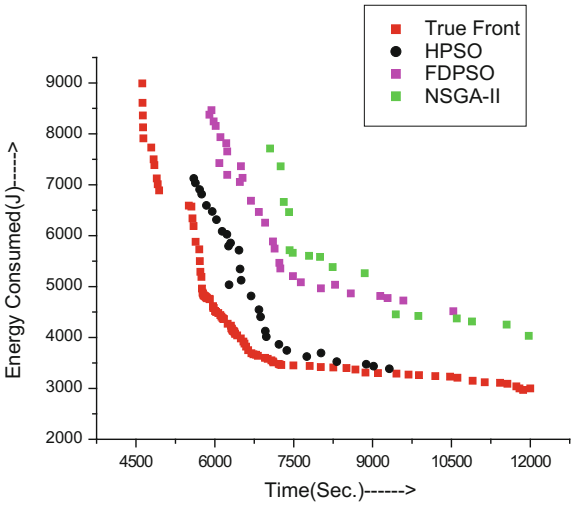


Fig. 7 Bi-objective non-dominated solutions for epigenomics workflow

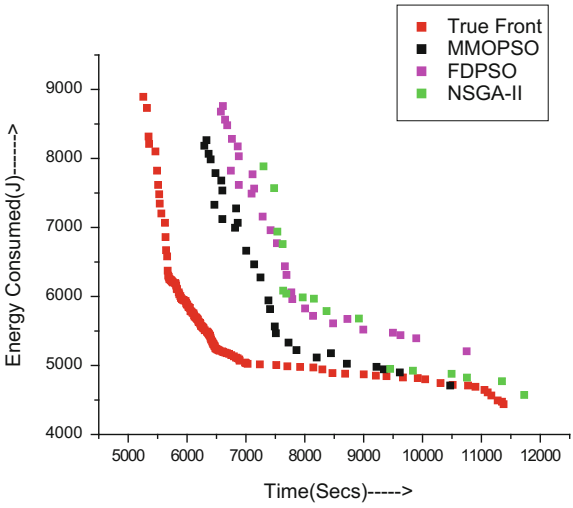
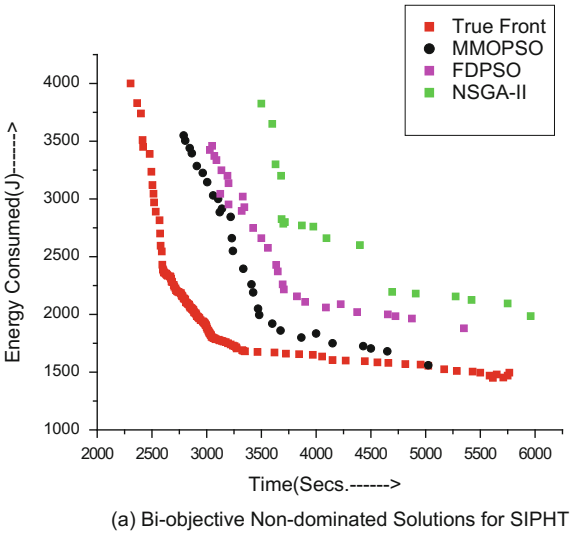


Fig. 8 Bi-objective non-dominated solutions for LIGO workflow

It has been observed from Table 3, the values of spacing metric using MMOPSO algorithm is 26, 43, 38, 36, and 40% lower than that of values of spacing metric obtained using FDPSO for Montage, CyberShake, Epigenomics, LIGO and SIPHT workflows, *respectively* as well as 35, 47, 34, 38, and 37% lower than that of values of spacing metric obtained using NSGA-II algorithm for Montage, CyberShake, Epigenomics, LIGO and SIPHT workflows, *respectively*. This is due to use of trade-off schedule plan between makespan and energy in the creation of non-dominated solution set. So, it is concluded that MMOHPSO algorithm provides uniform spacing as well as better convergence among the solution set as



(a) Bi-objective Non-dominated Solutions for SIPHT

Fig. 9 Bi-objective non-dominated solutions for SIPHT workflow

Table 2 Comparative results of GD for all workflow structures

Workflow	Generational distance (GD)		
	MMOPSO	NSGA-II	FDPSO
Montage	0.0016	0.1016	0.0544
CyberShake	0.0094	0.0396	0.0595
EpiGenomics	0.0097	0.0709	0.037
LIGO	0.0048	0.0358	0.0204
SIPHT	0.0047	0.0423	0.0206

Table 3 Comparative results of Spread for all workflow structures

Workflow	Spread		
	MMOPSO	NSGA-II	FDPSO
Montage	0.4736	0.7738	0.6033
CyberShake	0.2157	0.7819	0.7288
EpiGenomics	0.3612	0.5715	0.5426
LIGO	0.285	0.7041	0.6965
SIPHT	0.4315	0.7409	0.7128

compared to other algorithms for all workflow structures under consideration. Hence, it is applicable to offload large workflows task like face detection and matrix multiplication etc., over Mobile Cloud environment.

7 Conclusion and Future Work

In the past few years, a single objective task offloading problem has been addresses by many researchers. However, in real life applications, there are multiple conflicting objectives that must be satisfied simultaneously. So, the goal of decision maker is multi-fold and prefers the set of Pareto optimal solutions. To address this issue, we proposed the *Modified Multi-Objective Particle Swarm Optimization* (*MMOPSO*) algorithm based on the concept of dominance to solve the mobile cloud task scheduling problem. It is a combination of multi-objective particle swarm optimization algorithm and list based heuristic. Its performance is analyzed using two conflicting objectives of makespan, and energy consumption under application completion constraints. The efficacy and applicability of the proposed approaches are demonstrated by using different application task graphs and comparing it with state-of-art MOO techniques. The simulation experiments exhibit that MMOPSO performs better and generates the solutions that are more converged towards the true Pareto optimal front and shown uniform spacing among the created solutions. Hence, it is applicable to solve a wide class of multi-objective optimization problems for scheduling tasks over Mobile Cloud environment.

In future, the concept of neural networks, fuzzy logic, etc. needs to be tested for possible enhancement to the proposed heuristic for real life case studies.

References

1. R. Buyya, R. Buyya, C.S. Yeo, C.S. Yeo, S. Venugopal, S. Venugopal et al., Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Futur. Gener. Comput. Syst. **25**, 17 (2009). doi:10.1016/j.future.2008.12.001
2. C.J.R. Gabriel, M., Wolfgang G., Hybrid computing—where hpc meets grid and cloud computing. J Futur. Gener. Comput. Syst. **27**, 440–453 (2011)
3. S. Abrishami, M. Naghibzadeh, Deadline-constrained workflow scheduling in software as a service cloud. Sci. Iran. **19**, 680–689 (2012). doi:10.1016/j.scient.2011.11.047
4. R. Van Den Bossche, K. Vanmechelen, J. Broeckhove, Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds. Futur. Gener. Comput. Syst. **29**, 973–985 (2013). doi:10.1016/j.future.2012.12.012
5. S. Abrishami, M. Naghibzadeh, D.H.J. Epema, Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds. Futur. Gener. Comput. Syst. **29**, 158–169 (2013). doi:10.1016/j.future.2012.05.004
6. P. Parwekar, From Internet of Things towards cloud of things, in *2nd International Conference on Computer and Communication Technologies ICCCT-2011*, 2011, pp. 329–333. doi:10.1109/ICCCT.2011.6075156
7. A. Botta, W. De Donato, V. Persico, A. Pescape, On the integration of cloud computing and internet of things, in *International Conference on Future Internet Things Cloud, FiCloud 2014*, 2014, pp. 23–30. doi:10.1109/FiCloud.2014.14
8. A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of Cloud computing and Internet of Things: A survey. Futur. Gener. Comput. Syst. **56**, 684–700 (2016). doi:10.1016/j.future.2015.09.021

9. D. Gachet, M. De Buenaga, F. Aparicio, V. Padrón, Integrating internet of things and cloud computing for health services provisioning: the virtual cloud carer project, in *6th International Conference on Innovative Mobile Internet Services Ubiquitous Computing IMIS 2012*, 2012, pp. 918–921. doi:10.1109/IMIS.2012.25
10. R. Petrolo, V. Loscrí, N. Mitton, Towards a smart city based on cloud of things, in *ACM International Workshop Wireless Mobile Technologies Smart Cities—WiMobCity '14*, 2014, pp. 61–66. doi:10.1145/2633661.2633667
11. S.Y. Chen, C.F. Lai, Y.M. Huang, Y.L. Jeng, Intelligent home-appliance recognition over IoT cloud network, in *9th International Wireless Communications on Mobile Computing Conference IWCMC 2013*, 2013, pp. 639–643. doi:10.1109/IWCMC.2013.6583632
12. R.C. Andrea Prati, R. Vezzani, M. Fornaciari, Intelligent video surveillance as a service, in *Intelligent Multimedia Surveillance*, ed. by A.C. Pradeep, K. Atrey, M.S. Kankanhalli (Springer, Berlin, 2013), pp. 1–16
13. M. Yun, B. Yuxin, Research on the architecture and key technology of internet of things (loT) applied on smart grid, in *International Conference on Energy Science and Electrical Engineering (ICAEE)*, 2010: pp. 69–72. doi:10.1109/ICAEE.2010.5557611
14. B.B.P. Rao, P. Saluia, N. Sharma, A. Mittal, S.V. Sharma, Cloud computing for Internet of Things & sensing based applications, in *Sixth International Conference on Sensing Technology (ICST)* 2012, pp. 374–380. doi:10.1109/ICSensT.2012.6461705
15. C.M. Sarathchandra Magurawalage, K. Yang, L. Hu, J. Zhang, Energy-efficient and network-aware offloading algorithm for mobile cloud computing. Comput. Netw. **74**, 22–33 (2014). doi:10.1016/j.comnet.2014.06.020
16. M.Y. Wu, T Haluk, H. Salim, Performance-effective and low-complexity task scheduling for heterogenous computing. IEEE Trans. Parallel Distrib. Syst. **13**, 260–274 (2002)
17. S.C. Kim, S. Lee, J. Hahm, Push-pull: deterministic search-based DAG scheduling for heterogeneous cluster systems. IEEE Trans. Parallel Distrib. Syst. **18**, 1489–1502 (2007). doi:10.1109/TPDS.2007.1106
18. L. Yang, J. Cao, S. Tang, T. Li, A.T.S. Chan, A framework for partitioning and execution of data stream applications in mobile cloud computing, in *IEEE 5th International Conference* on *Cloud Computing CLOUD 2012*, 2012, pp. 794–802. doi:10.1109/CLOUD.2012.97
19. E. Ilavarasan, R. Manoharan, High performance and energy efficient task scheduling algorithm for heterogeneous mobile computing system. J. Comput. Sci. **2**, 10–27 (2010)
20. Z. Tang, L. Qi, Z. Cheng, K. Li, S.U. Khan, K. Li, An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment. J. Grid Comput. **14**, 55–74 (2016). doi:10.1007/s10723-015-9334-y
21. M. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, R. Govindan, Odessa : enabling interactive perception applications on mobile devices categories and subject descriptors, in *Proceedings of the International Conference* on *Mobile Systems*, *Applications*, and *Service*, 2011, pp. 43–56. doi:10.1145/1999995.2000000
22. P. Rong, M. Pedram, Power-aware scheduling and dynamic voltage setting for tasks running on a hard real-time system, in *Asia and South Pacific Design Automation Conference*, 2006, pp. 473–478. doi:10.1109/ASPDAC.2006.1594730
23. Z. Li, C. Wang, R. Xu, Task allocation for distributed multimedia processing on wirelessly networked handheld devices, in *16th IEEE International Parallel & Distributed Processing Symposium*, 2001, pp. 741–746. doi:10.1109/IPDPS.2002.1015589
24. K. Kumar, Y. Lu, Cloud computing for mobile users : computation save energy ? Computer (Long. Beach. Calif) **43**, 51–56 (2010)
25. Y.C. Lee, A.Y. Zomaya, A novel state transition method for metaheuristic-based scheduling in heterogeneous computing systems. IEEE Trans. Parallel Distrib. Syst. **19**, 1215–1223 (2008). doi:10.1109/TPDS.2007.70815
26. D. Kovachev, T. Yu, R. Klamma, Adaptive computation offloading from mobile devices into the cloud, in *10th IEEE International Symposium on Parallel and Distributed Processing and Applications ISPA 2012*, 2012: pp. 784–791. doi:10.1109/ISPA.2012.115

27. W. Huijun, H. Dijiang, S. Bouzefrane, Making offloading decisions resistant to network unavailability for mobile cloud collaboration, in: *International Conference on Collaborative Computing*: *Networking*, *Applications* and *Worksharing (Collaboratecom), 2013*, pp. 168–177. doi:10.4108/icst.collaboratecom.2013.254106
28. S. Ou, K. Yang, L. Hu, CRoSS: a combined routing and surrogate selection algorithm for pervasive service offloading in mobile ad hoc environments, in *GLOBECOM—IEEE Global Telecommunications Conference,* 2007, pp. 720–725. doi:10.1109/GLOCOM.2007.140
29. L.H.L. Huang, Q.X.Q. Xu, Energy-efficient task allocation and scheduling for multi-mode MPSoCs under lifetime reliability constraint,in *Design, Automation & Test* in *Europe Conference & Exhibition*, *(DATE)*, 2010. doi:10.1109/DATE.2010.5457063
30. K. Sinha, M. Kulkarni, Techniques for fine-grained, multi-site computation offloading, in *IEEE/ACM International Symposium on Cluster*, *Cloud*, and *Grid Computing CCGrid 2011*, 2011, pp. 184–194. doi:10.1109/CCGrid.2011.69
31. X. Lin, Y. Wang, Q. Xie, M. Pedram, Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. IEEE Trans. Serv. Comput. **8**, 175–186 (2015). doi:10.1109/TSC.2014.2381227
32. M. Shiraz, A. Gani, A. Shamim, S. Khan, R.W. Ahmad, Energy efficient computational offloading framework for mobile cloud computing. J. Grid Comput. **13**, 1–18 (2015). doi:10.1007/s10723-014-9323-6
33. M. Akram, A. Elnahas, Energy-aware offloading technique for Mobile cloud computing, in *Proceedings of the 2015 International Conference on Future Internet* of *Things and Cloud*, *FiCloud 2015*. *International Conference* on *Open* and *Big Data, OBD 2015*, 2015, pp. 349–356. doi:10.1109/FiCloud.2015.45
34. O.W. Samuel, G.M. Asogbon, A.K. Sangaiah, P. Fang, G. Li, An integrated decision support system based on ANN and Fuzzy_AHP for heart failure risk prediction. Expert Syst. Appl. Elsevier Publishers **68**, 163–172 (2017)
35. A.K. Sangaiah, A.K. Thangavelu, X.Z. Gao, N. Anbazhagan, M.S. Durai, An ANFIS approach for evaluation of team-level service climate in GSD projects using Taguchi-genetic learning algorithm. Appl. Soft Comput. **30**, 628–635 (2015)
36. A.K. Sangaiah, A.K. Thangavelu, An adaptive neuro-fuzzy approach to evaluation of team-level service climate in GSD projects. Neural Comput. Appl. Springer Publishers **23**(8) (2013). doi:10.1007/s00521-013-1521-9
37. C.C. Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization. IEEE Trans. Evol. Comput. **8**, 256–279 (2004). doi:10.1109/TEVC.2004.826067
38. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**, 182–197 (2002). doi:10.1109/4235.996017
39. R. Garg, A.K. Singh, Multi-objective workflow grid scheduling using ε-fuzzy dominance sort based discrete particle swarm optimization. J. Supercomput. **68**, 709–732 (2014). doi:10.1007/s11227-013-1059-8
40. A. Verma, S. Kaushal, Cost-time efficient scheduling plan for executing workflows in the cloud. J. Grid Comput. **13**, 495–506 (2015). doi:10.1007/s10723-015-9344-9
41. M. Mezmaz, N. Melab, Y. Kessaci, Y.C. Lee, E.G. Talbi, a. Y. Zomaya, et al., A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems, J. Parallel Distrib. Comput. **71**, 1497–1508 (2011). doi:10.1016/j.jpdc.2011.04.007
42. K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*(Wiley, 2005), pp. 13–46
43. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of the IEEE International Conference on Neural Networks, 1995*, vol. 4, pp. 1942–1948. doi:10.1109/ICNN.1995.488968
44. P.K. Tripathi, S. Bandyopadhyay, S.K. Pal, Multi-Objective Particle Swarm Optimization with time variant inertia and acceleration coefficients. Inf. Sci. (NY) **177**, 5033–5049 (2007). doi:10.1016/j.ins.2007.06.018

45. R. Garg, Multi-objective optimization to workflow grid scheduling using reference point based evolutionary algorithm. Int. J. Comput. Appl. **22**, 44–49 (2011)
46. S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, S Lanitchi, K. Vahi, et al., Characterization of scientific workflows, in *Workflows in Support of Large-Scale Science, CA, USA, 2008*, pp. 1–10. doi:10.1109/WORKS.2008.4723958
47. R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**, 23–50 (2011). doi:10.1002/spe.995