A Discrete Particle Swarm Optimization Approach for Energy-Efficient IoT Services Placement Over Fog Infrastructures

Conference Paper · June 2019

DOI: 10.1109/ISPDC.2019.00020

Tanissia Djemai
Paul Sabatier University - Toulouse III
1 PUBLICATION 0 CITATIONS
SEE PROFILE

Patricia Stolf
Institut de Recherche en Informatique de Toulouse
69 PUBLICATIONS 458 CITATIONS
SEE PROFILE

Thierry Monteil
Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS)
107 PUBLICATIONS 717 CITATIONS
SEE PROFILE

Jean-Marc Pierson
Paul Sabatier University - Toulouse III
189 PUBLICATIONS 1,589 CITATIONS
SEE PROFILE

A Discrete Particle Swarm Optimization approach for Energy-efficient IoT services placement over Fog infrastructures

*

Tanissia Djemai
IRIT, LAAS-CNRS
University of Toulouse
Toulouse, France
tanissia.djemai@irit.fr

Patricia Stolf
IRIT
University of Toulouse
Toulouse, France
patricia.stolf@irit.fr

Thierry Monteil
LAAS-CNRS
University of Toulouse, CNRS, INSA
Toulouse, France
monteil@laas.fr

Jean-Marc Pierson
IRIT
University of Toulouse
Toulouse, France
jean-marc.pierson@irit.fr

***Abstract*—The Internet of Things (IoT) encompasses both large-scale deployed physical infrastructures and software layers that enable intuitive and transparent creation of applications. This highly distributed, energy-greedy environment must ensure the quality of deployed services while taking into account the heterogeneity of capabilities and protocols as well as users and objects mobility. Deployment infrastructure has been redesigned to provide the necessary features, including paradigms such as software-defined networks and Fog computing. The purpose of this article is to study IoT services placement in a Fog architecture. We propose a model of the infrastructure and IoT applications as well as a placement strategy taking into account system's energy consumption and applications delay violations minimization with a Discrete Particles Swarm Optimization algorithm (DPSO). Simulations have been done with iFogSim simulator. Results have been compared with heuristics coming from the literature: Binary Partical Swarm optimization (BPSO), Dicothomous Module Mapping (DCT), CloudOnly, IoTFogOnly, IoTCloud (IC) and FogCloud (FC) placement approaches.**

***Index Terms*—Fog, IoT, DPSO, Heuristics, Energy, QoS, iFogSim.**

I. Introduction

Nowadays, we depict around 15 billions of deployed connected objects. A study published by CISCO [25] predicts an increase of devices reaching 50 billions in 2025. The proliferation of equipment such as smartphones, wearables, autonomous vehicles and their associated services leads to high heterogeneity, scalability, delays and mobility issues and that is making Internet of Things (IoT) services placement in the cloud less attractive. Fog Computing is defined as the process of extending cloud capabilities with networks nodes [1], [2] and seems to be a promising solution to support requirements induced by IoT applications. Indeed, Fog computing has emerged as a powerful paradigm answering new application needs such as short response delays and high bandwidth demand by providing computing and storage capacities closer to end users. Adding to that, a large scalable and distributed infrastructure such as Fog computing is at first sight more suitable to handle objects heterogeneity and dynamic of IoT applications but in contrast, their placement and management in such infrastructure can be quite challenging and disadvantageous if not well designed. Using such a distributed large scale and dynamic topology can be easily energy and network-greedy if the trade-off between services Cloud-placed, Fog-placed is not well balanced. Adding to that, objects and users positions are also parameters that should be considered for IoT services, which also interact with real physical environment through sensors and actuators devices. Energy consumption is a high interest field for cloud community, considering that data centers consume 4% of global world energy [27]. Fog Computing, by combining both data centers, communications infrastructures and IoT systems makes the energy issue more complex to study, inheriting various issues of evaluation and optimization of the latter in networks and IoT systems. A report from the International Energy Agency (IEA) [11] estimates the annual energy consumption of 5 types of IoT applications, home automation, smart lighting and smart street lighting, smart roads, smart appliances, to 46 TWh in 2025. Since services placement problem is NP-hard, we propose to use an evolutionary semi-stochastic meta-heuristic to efficiently place IoT services in a Fog infrastructure minimizing the energy consumption and applications delay violations while ensuring nodes capacities constraints and services placement constraints. This paper's contributions can be summarized as follows :

- Using a Discrete Particle Swarm Optimization for IoT services placement in a Fog computing infrastructure.
- Considering different applications topologies and classes with their delay constraints.
- Evaluating with real topology values and different networks technologies using iFogSim simulator.

The remainder of this paper is structured as follows. In section II we establish a brief Fog computing and services placement's state of the art. Then, we introduce Particle Swarm approach. In section III and IV, we expose respectively the problem modeling and the Discrete Particle Swarm Optimization (DPSO) model used for IoT services placement. Finally, section V reports experimental results using iFogSim.

II. Related work

A. Fog computing and services placement

Fog Computing is a relatively young concept [1]. Since 2012, several Fog research areas have emerged, including the development of simulation tools such as iFogSim and EdgeCloudSim [3] [23], platforms for the industry, such as Iox [5] and Edgex [23]. [14] determines the degree of relevance of the Fog according to the type of applications and their use cases: connected vehicles, health care, smart tracking, smart grids, [29] defines classes of services to facilitate deployment policies. There are also many works on Fog-Cloud and Fog-IoT layers interactions, moving loads from a Fog node to the Cloud, assigning users to Cloud and Fog services, increasing capacity mobile devices as part of Mobile Cloud Computing (MCC) [31] and Mobile Edge Computing (MEC) [30]. [15], deals with the offloading of Fog tasks to the Cloud under delay constraints. Other works like [16] tackle the issue of fog node allocation to different service providers by combining principles of game theory with Stackelberg's bidding.

Several approaches for IoT service placement in the Fog have been addressed. [7] presents a generic placement algorithm of IoT services in the Fog based on a dichotomous search on all the available Fog nodes. [12] proposes a method which identifies deployment's eligibility, taking into account business policies and resources states. [17] proposes to maximize the placement of tasks in the Fog without taking into account the requirements of services and excludes cloud usage in certain cases. Following the same objective of fog resource usage maximization [13] proposes a genetic algorithm approach. [18] studies scheduling problem in Fog computing, Considering user mobility influence on application performance and compares three different scheduling policies, namely concurrent, FCFS, and shows that delay-priority strategies, can be used to improve execution [19] try to minimize application makespan and monetary cost.

About energy in Fog Computing and IoT, [4] defines elements that should come into consideration to reduce cloud-IoT system energy consumption such as network access technology, applications type, Fog servers energy consumption with no workload, virtualization techniques and their management. [24] shows that IoT applications energy consumption in the Fog is directly impacted by access technologies. It is shown that application can be more energy-greedy if it is accessed via a 4G network rather than a wired network with Ethernet technology, sending rates also influences, and in some cases wired technologies are less efficient than wireless. [21] proposes a strategy of clustering Fog nodes to minimize their energy consumption. [8] tries to minimize energy, time and execution cost for mobiles tasks; it studies offloading rather than initial services placement and model the problem with queues system modeling. In [20], model for video traffic's energy consumption in a Fog system is proposed but services placement is supposed to be already done. [22] adopted an object assignment problem rather than service one by addressing IoT devices to Fog nodes assignment problem with an evolutionary algorithm to reduce mobile energy consumption under delay constraints. Due to computing and networking capacities evolution, agent-based evolutionary algorithms are receiving more attention and have emerged as strong optimization approaches. Meta-heuristics such as Genetic Algorithm (GA), Ant Colony Optimization (ACO) or Particle Swarm Optimization (PSO) seem to be the ideal approaches for distributed and dynamic computing systems. To the best of our knowledge, Particle Swarm Optimization was only used in cloud paradigm. [40] used PSO for task assignment problem in order to reduce total execution time and compare PSO results with classical GA. In the same field, [42] proposes a DPSO approach for grid job scheduling aiming to minimize makespan and flowtime and shows that this approach gives better results than other evolutionary methods such as GA and ACO. [43] proposes an energy-efficient routing protocol for Wireless network using PSO. Differently from previous works and considering good results given by the PSO approach in cloud paradigm, we aim to place different types of IoT applications in a Fog infrastructure while minimizing both system's energy consumption and applications delay violations using a Discrete Particles Swarm (DPSO) metaheuristic under devices capacities constraints. To the best of our knowledge, there is no other work in the literature that aims at minimizing both energy consumption and applications delay violations.

B. Particle Swarm Optimization approach overview

Particle Swarm approach [38] is a semi-stochastic population-based optimization method inspired by the collective behavior of social animals such as flock of birds and fish school. In order to find an acceptable solution to a combinatorial problem, the PSO manipulates a population of particles set called a Swarm. This particle swarm explores the problem search space to find an acceptable solution.

Considering a D-dimensional combinatorial continuous problem, the k^{th} swarm's particle $\vec{X_k}$ is a D-dimensional vector representing a feasible solution. Each particle $\vec{X_k}$ is identified by a subset of particles' neighbours, its position in the search space $\vec{X_k}^t$ and its motion speed called velocity $\vec{V_k}^t$ which varies from one iteration t to the next. The speed variation of each particle $\vec{X_k}$ is a function of particle's previous position $\vec{X_k}^{(t-1)}$, its personal best known position $\vec{Pb_k}$ and its neighbours best position $\vec{Nb_k}$ (eq (1)).

$$\vec{X_k}^t = f(\vec{X_k}^{t-1}, \vec{V_k}^{t-1}, \vec{Pb_k}, \vec{Nb_k}) \quad (1)$$

PSO was initially proposed for continuous problems but rapidly, in order to resolve binary decision problems such as services placement, Kennedy and Eberhart introduced Binary Particle Swarm Optimization (BPSO) [36]. Its main difference with PSO lies in particles positions and velocities definitions which are defined respectively as binary placement and probabilities matrices. For this purpose, the sigmoid function $sig(.)$

is introduced to map all real valued elements of velocity $v_k^t(i,j)$in $[0,1]$ with $i,j \in \mathbb{N}$.

- $\forall X_k$ particle of the swarm and its corresponding velocity V_k, defined respectively as a binary placement and probability matrices, are updated according to equations (2), (3) and (4) as follows:

$$V_k^{t+1} = \omega V_k^t + \varphi_1 \omega_1^t (Pb_k^t - X_k^t) + \varphi_2 \omega_2^t (Nb_k^t - X_k^t) \quad (2)$$

$$x_k^{t+1}(i,j) = \begin{cases} 1 & \text{if rand()} \geq sig(v_k^{t+1}(i,j)) \\ 0 & \text{if rand()} < sig(v_k^{t+1}(i,j)) \end{cases} \quad (3)$$

$$sig(v_k^{t+1}(i,j)) = \frac{1}{1 + \exp^{-v_k^{t+1}(i,j)}} \quad (4)$$

- φ_1 and φ_2 are respectively known as cognitive and social constants that modulate the magnitude of particle's next step to its personal best and neighbours best solutions. According to the literature [37] φ_1 and φ_2 are usually in $[0,4]$ and it has been shown in [38] that $\varphi_1 = \varphi_2 = 2$ works well for most applications.
- To avoid particle's big oscillations problem, literature introduces two approaches: Velocity clamping that bounds the velocity vector elements $v_k(i,j) \in [-V_{max}, V_{max}]$ or a restriction coefficient ω to ponderate velocity elements values. This limitation prevents particles from moving too rapidly from one search space region to another. This value is usually initialized as a function of the problem range [35], [37].
- ω_1 and ω_2 are two matrices with elements $\omega_1(i,j), \omega_2(i,j)$ taken randomly in $[0,1]$, aiming to introduce randomness in particles motions behaviour through search space.

This approach was designed for discrete problems as DPSO where values of X_k are in $\mathbb{N}$. For placement problems DPSO, that we will detail in section IV, is more interesting in term of memory space usage during implementation and we can easily switch from BPSO to DPSO modeling. [42].

III. Problem Formulation

A. Physical topology

We consider a Hierarchical Three-Layered Fog infrastructure $\mathcal{M}$, constituted of M physical nodes. The first layer is composed of a set $\mathcal{C}$ of cloud data center's nodes (level 0), the second one is a set $\mathcal{F}$ of Fog nodes (level 1 to $l-1$) and the last layer regroups a set $\mathcal{T}$ of IoT devices or connected things (level l) to which sensors and actuators are directly connected. The physical topology is represented by an oriented graph $G_{\mathcal{M}} = (\mathcal{M}, \mathcal{L})$ with $\mathcal{L}$ the set of links between nodes. Each node $m_i \in \mathcal{M}$ with $i \in [0, M-1]$ has the following characteristics:

- A Level lev_i in the topology from 0 to l.
- Processing capacity cpu_i in MIPS.
- Memory capacity ram_i in MB.
- Power consumption characteristics p_i^{idle} that represents the power consumption of the device when it is not used and p_i^{max} the device's power consumption when it is used to its maximal capacity.

Each link $l_k \in \mathcal{L}$ that binds nodes m_i and m_j has the following characteristics:

$$k = \begin{cases} (i,j) \ with \ i,j \in [0, M-1] \text{ If point to point} \\ (i, j_1, .., j_n) \ with \ j_n \in [0, M-1], n \in \mathbb{N} \text{ If multi-points} \end{cases}$$

- ntw_k represents links network technology.
- bw_k is the bandwidth in Mb/s.
- lc_k is the latency in ms.
- ste_k is the state which can be "On" or "Off".

l_k can be a physical or a virtual link. If l_k is a virtual link, we assume that devices' routing algorithm gives us one optimum path between two machines composed of a set Θ_{l_k} of physical links.

$\forall v \in \Theta_{l_k}$

- $bw_k = \min_{v \in \Theta_{l_k}}(bw_v)$
- $lc_k = \sum_{v \in \Theta_{l_k}}(lc_v)$

B. IoT Applications

According to works [28], [10] and [29], IoT applications are special workflows that should be deployed in distributed systems and can be modeled as Directed Acyclic Graphs (DAG). Indeed, an IoT application gets data from the physical environment through sensors equipment, those information are processed by software services and then instructions are transmitted to actuators in order to act on devices. From the previous definition we can decompose the IoT application view as follows:

1) *Sensors and Actuators:* In application's DAG representation, sensors nodes and actuators nodes represent respectively sources and ends of the DAG data flow.

2) *Processing services:* Each processing service s_i is defined by:

- tec_i: the service deployment technology which can be a virtual machine, a container, OSGi plugin, Java Virtual Machine(JVM) or a combination of them.
- mi_i: the CPU requested by service i in million instructions.
- ram_i: the maximum RAM requested by service i in MB.
- d_i^{exe}: the maximum execution delay for service i in ms.

3) *Links and dependencies between IoT processing services:* Each oriented edge $e_{(i,j)} \in \mathcal{E}$ going from service s_i to service s_j represents data dependency between s_i and s_j and carries the following information:

- $data_{(i,j)}$ is the data size sent from s_i to s_j in kB.
- $d_{(i,j)}^{com}$ is the maximum communication delay between s_i and s_j.

4) *IoT application Direct Acyclic Graph models:* We consider a Directed Acyclic Graph $G_{\mathcal{A}} = (\mathcal{S}, \mathcal{E})$ which represents a set $\mathcal{A} = \{a_0, ..., a_{A-1}\}$ of IoT applications where each application a_i has a size n_i of processing services.

$\mathcal{S}$ is the set of size N composed of processing services from all applications that we have to place with $N = \sum_{i=0}^{A-1} n_i$.

$\mathcal{E}$ is the set of edges representing data dependencies between the applications graph nodes.

5) IoT applications service classes and priority: Establishing a prioritization between applications classes shows usually good results for standard workflows management in cloud data centers [29]. Each IoT application belongs to a class as identified in [9] with its associated priority. In addition to services' execution time and communication delay, each application a_i has a global maximum response delay $D_{a_i}^{max}$, from a sensor to its corresponding actuator, that should not be exceeded. This value is taken according to delay-sensitivity in each application class.

C. Problem statement

We aim to place a set $\mathcal{A} = \{a_0, a_1, ..., a_{A-1}\}$ of heterogeneous IoT applications in a three-Layered Fog infrastructure's nodes $\mathcal{M} = \{m_0, ..., m_{M-1}\}$ while minimizing a cost function f under nodes capacities constraints.

$f : \mathbb{N}^N \longmapsto \mathbb{R}$ represents system's total energy consumption E_T pondered by total applications delay violations λ.

The total applications delay violations λ, defined in eq (5), represents the count of all sensor to actuator delay violations in the applications set $\mathcal{A}$.

A delay violation of an application a_i occurs when the processing and communication time d_{a_i}, from its sensor to its attached actuator exceeds the allowed time $D_{a_i}^{max}$ which is given by the application's class requirements.

$$\lambda = \sum_{a_i \in \mathcal{A}} w_{a_i}$$
with :
$$w_{a_i} = \begin{cases} 1 & \text{if } D_{a_i}^{max} < d_{a_i} \\ 0 & \text{if not} \end{cases} \quad (5)$$

The total energy consumption E_T is the sum of energy consumption from the computations and from network communications.

$$E_T = E_N + E_C. \quad (6)$$

For one data flow processing, i.e from sensor to actuator, we compute energy consumption as follow:

$$E_C = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} Y_k^t(i,j) \frac{mi_i}{cpu_j} (p_j^{max} - p_j^{idle}) + p_j^{idle} \quad (7)$$

$$E_N = \sum_{(i,l)\in\mathcal{S}} \sum_{(j,p)\in\mathcal{M}} [Y_k^t(i,j) Y_k^t(l,p) \frac{data_{(i,l)}}{bw_{(j,p)}} \quad (8)$$
$$+ lc_{(j,p)}][p_j^{max} + p_p^{max} - p_j^{idle} - p_p^{idle}]$$

with $Y_k^t(i,j)$ is the binary decision variable for the placement of service s_i in the device m_j from the decision agent[1] k at time t.

[1]Swarm based algorithms manipulate a set of potential solutions that evolve during algorithm life time. Depending on the idea of the swarm approach, the terminology of this potential group of solutions varies from one approach to another. To give a method-agnostic naming to a swarm single element we use "decision agent" term.

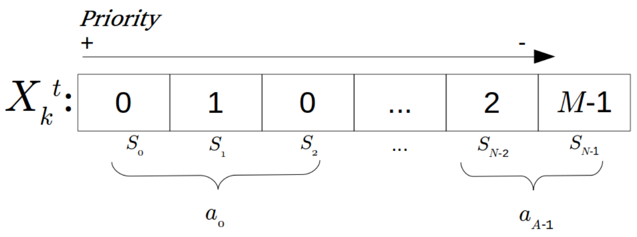


Fig. 1: **Position of the k^{th} particle after t iterations.**

$\forall k$ in decision agents set and $\forall t \in \mathbb{N}$

$$Y_k^t(i,j) = \begin{cases} 1 & \text{if service } s_i \text{ is on machine } m_j \\ 0 & \text{if not} \end{cases} \quad (9)$$

The problem is equivalent to place the set of all services $\mathcal{S} = \{s_0, s_1, ..., s_{N-1}\}$ of $G_{\mathcal{A}=(\mathcal{S},\mathcal{E})}$, which is the DAG of all applications to place at the same time, on physical topology nodes set $\mathcal{M}$.

$\forall k, t \in \mathbb{N}$:

$$f = min_{i\in[0,N-1], j\in[0,M-1]}[(1+\lambda)E_T] \quad (10)$$

$$s.t \begin{cases} ram_i \leq ram_j \forall i \in [0, N-1], \forall j \in [0, M-1]..(i) \\ mi_i \leq cpu_j \forall i \in [0, N-1], \forall j \in [0, M-1]..(ii) \\ \sum_{j\in[0,M-1]} y_k^t(i,j) = 1, \forall i \in [0, N-1]...(iii) \end{cases}$$

- (i) and (ii) are respectively memory and computing constraints for placing service i on machine j.
- (iii) means that a service s_i should be placed only in one device.

IV. A DISCRETE PSO FOR IOT SERVICES PLACEMENT

We have a particles swarm $\mathcal{P} = \{X_0, ..X_{P-1}\}$ of size P, each $X_k \in \mathcal{P}$ will evolve over a set T_{max} of iterations.

A. Particle position and velocity's representations

As it is shown in Figure 1, $X_k \in \mathbb{N}^N$ represents the position of the k^{th} particle in the DPSO swarm and it is a N size vector with values $x_k(i) \in [0, M-1], \forall i \in [0, N-1]$. X_k^t is the particle's position at iteration t. $x_k^t(i) = z$ means that at iteration t service s_i is placed in the z^{th} machine m_z with $z \in [0, M-1]$. Which can be written as follows: $x_k^t(i) = z \Longleftrightarrow Y_k^t(i,z) = 1 \wedge \forall j \in \{0, .., N-1\} - \{z\}, Y_k^t(i,j) = 0$. The particle's velocity V_k is an NxM matrix. It determines the motion speed of the particle X_k. Each element $v_k(i,j) \in \mathbb{R}$ defines the possibility of service s_i to be placed in the machine m_j.

B. Particle's motion equation

Particle positions are updated through iterations according to the following:

1) We compute each particle new velocity matrix according to Eq (11).

$$v_k^{t+1}(i,j) = \omega^{(t+1)} v_k^t(i,j) + \varphi_1 \omega_1^{t+1}(i,j)[f(Pb_k^t) - f(X_k^t)] + \varphi_2 \omega_2^{t+1}(i,j)[f(Nb_k^t) - f(X_k^t)] \quad (11)$$

We may stress that variables $\omega_1, \omega_2, \varphi_1, \varphi_2, \omega, Nb, Pb$ keep the same meaning as in the BPSO equation (2). The difference lies in the particle representation which is a vector rather than a binary matrix and this representation is better to use less memory space with vector data structures. Adding to that, considering the saturation problem of the sigmoid function in the BPSO, we prefer to use real valued velocities matrices V_k. We have also introduced the fitness into velocity updating equation rather than using particle vector index. ω modulates the influence of the previous speed in the new speed computation. According to literature [37], [38], [35] its value should vary in $[\omega_{min}, \omega_{max}]$ with $\omega_{min} = 0.4$ and $\omega_{max} = 0.9$ and the linear decrease strategies have shown best results for algorithm stabilization. For our strategy, we have chosen to update ω according to [39] as follows:

$$\omega^t = \omega_{max} - \frac{(\omega_{max} - \omega_{min})}{T_{max}} * t \tag{12}$$

2) After new velocity computation, we deduce particle's new position vector with Eq (13)

$$x_k^t(i) = Z \iff v_k^t(i, Z) = \max_{\forall j \in [0, M-1]} \{v_k^t(i, j)\} \tag{13}$$

3) Physical topology constraints can reduce placement possibilities for a service s_i. Each service $s_i \in \mathcal{S}$ has its authorized subset of physical machines. To ensure this constraint in the DPSO, if a service s_i can't be placed in a device $m_j \in \mathcal{M}$:

$$\forall k \in [0, P-1], \forall t \in [0, T_{max} - 1] \Rightarrow v_k^t(i, j) = -\infty \tag{14}$$

C. Initial population

The initial swarm particles $X_0^0, .., X_{P-1}^0$ are distributed uniformly in search space.

$\forall s_i \in \mathcal{S}$, $x_k^0(i)$ should be taken uniformly in service s_i admissible set of machines.

Initial velocities V_k^0 are initialized to 1 and $-\infty$ elements matrices as described in Algorithm 2.

D. Neighboring Topology

Neighbourhood topology defines particles swarm communications. In its original version [38], PSO algorithm allows all particles to exchange their solutions to determine a global best solution. This global approach leads particles to be trapped in a local optimum. To avoid this problem, the swarm can be divided into sub-groups and communications will be allowed only between particles in the same sub-group [41]. According to literature, we can define geometrical sub-groups which are deduced based on closest particles considering a certain metric (e.g fitness function) or we can define social sub-groups that are defined based on particles swarm index and this last has shown better results than the geometrical approach [37], [38], [41]. Based on that, we define a social circular neighbourhood of two particles as shown in Figure 2.

We present the overall DPSO approach in Algorithm 1

```
Data:   · Topology nodes set $\mathcal{M}$ of size $M$
        · Applications services set $\mathcal{S}$ of size $N$
        · $P$, $T_{max}$, $\varphi_1$, $\varphi_2$, $\omega^0$,$\omega_1^0$,$\omega_2^0$
Result: · Best placement particle vector $G_{best}$
        · Total system energy $E_T$
        · Total applications delay violations $\lambda$
begin
    V$^0$, $P^0 = uniformInit()$
    t=0
    $\omega^0 = 0.9$
    while $t < T_{max}$ do
        for $X_k^t \in P^t$
         do
            updateVelocity($X_k^t$)
            updatePosition($X_k^t$)
            if      $(f(X_k^t) < f(Pb_k))$
              $Pb_k \leftarrow X_k^t$
            for $X_w \in Neighbors(X_k)$ and $w \in [0, P-1]$
             do
                if      $(f(X_k^t) < f(Nb_w))$
                $Nb_w \leftarrow X_k^t$
            end
        end
        $(Gbest \leftarrow X_z^t) \iff f(X_z^t) =$
         $min_{\forall k, z \in [0, P-1]}\{f(X_k^t)\}$
        t++;
    end
end
```

Algorithm 1: DPSO for services placement

```
Data:   · Topology size $M$
        · Services size $N$
        · Population size $P$
Result: · Initial particles Swarm $\mathcal{P}^0$
        · Initial velocities set $\mathcal{V}^0$
begin
    for $k \in [0, P-1]$
     do
        for $i \in [0, N-1]$
         do
            choose uniformly $j \in [0, M-1]$ with $m_j$
              allowed device for service $s_i$
            $x_k^0(i) \leftarrow j$
        end
        for $i \in [0, N-1]$ do
            for $j \in [0, M-1]$ do
                if      $m_j$ is an allowed device for $s_j$
                $v_k^0(i, j) = 1$
                elsif
                $v_k^0(i, j) = -\infty$
            end
        end
        $Pb_k \leftarrow X_k^0$
        for $X_w \in Neighbors(X_k)$ and $w \in [0, P-1]$
         do
            $Nb_w \leftarrow X_k^t$
        end
    end
    $(Gbest \leftarrow X_z^0) \iff f(X_z^0) =$
     $min_{\forall k, z \in [0, P-1]}\{f(X_k^0)\}$
end
```

Algorithm 2: Swarm and velocities initialization

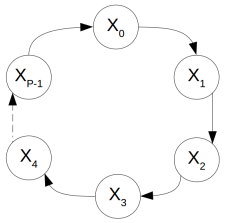


Fig. 2: Neighbouring topology.

V. Experimental Results

A. Test methodology

In order to prove the efficiency of the proposed method and to observe the impact of different Fog Layers-interplay strategies on both delay and energy consumption, the DPSO has been compared to BPSO, CloudOnly, IoTFogOnly, IoTCloud (IC), FogCloud (FC) and Dicothomous Modules Mapping (DCT) placement heuristics. Those methods are implemented in JAVA in iFogSim [3] which is a Fog environment simulator based on CloudSim tool [34]. Experiments have been conducted on an Intel core i7-7700 CPU@3.60GHz x8. For a given fixed Fog topology, we vary applications set size and compare the different approaches considering system's total energy consumption in MJoules and applications delay violations number as a QoS metric.

B. Strategies

We briefly define the implemented placement approaches as follows: (1) **CloudOnly** - For a placement of all services in cloud nodes. (2) **IoTFogOnly** - IoTFogOnly tries to place all services from the same application, under capacity and dependency constraints, in IoT closest node to application's user device then it moves to Fog layer closest node. (3) **FogCloud (FC)** - FogCloud places as much as possible services from same application in the same Fog layer node, under dependency and capacity constraints, then it uses cloud layer placement. (4) **IoTCloud (IC)** - It places as much as possible services in the IoT layer, under dependency and capacity constraints, then it uses cloud nodes. (5) **Dicothomous Module Mapping (DCT)** - This method is coming from [7], it sorts services and nodes respectively by the increasing order of their computation needs and computation capacities then a Dicothomous search over nodes is applied for each service. We adapted this strategy to respect services data dependencies. These algorithms are the baseline for the comparison with BPSO and DPSO. No other work in the literature minimizes both energy and applications delays violations. (6) **BPSO** - Is the implementation of Binary Particle Swarm algorithm as described in section II. (7) **DPSO** - For the proposed DPSO, Algorithm 1, the following ranges of parameter values have been tested: φ_1, φ_2 in $[1, 4]$ and $P \in [10, 80]$. Based on our pre-experiments results, which are in adequacy with values found in literature [37], [42], DPSO performs its best under the following settings: $\varphi_1 = 2$, $\varphi_2 = 2$ $P = 40$.

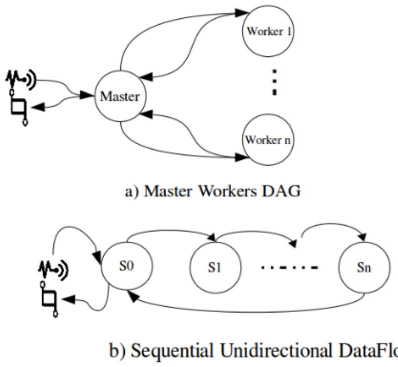


Fig. 3: DAG structures used for IoT applications generation

C. Input Data

1) A varying set of IoT applications: We consider in our work and experimentation, two types of Graphs that were given in [28] and shown in Figure 3.

- Master-Worker: The Master is the only service that communicates with the source and then sends data instructions to workers services. After the response, the master sends orders to actuators.
- Sequential Unidirectional DataFlow (SUD): It represents a sequential data flow from one service to another one. The first service gets data from sensors; the last one communicates with the actuators.

We place A applications with $A \in [4, 32]$ and a step of $h = 4$. Each application a_i has a size $n_i = 3$ of services. We took average services workload in Million instructions demand and network data size (kB) exchanged between them proportionally to information found in applications examples given in [28]. Each application a_i is characterized by its DAG structure, Quality Of Service (QoS) requirements and priority. We used table I to generate a set of IoT applications of different classes that have been identified in [9] with their associated priority (0 is the highest priority) and QoS requirements. 50% of master-slave and 50% of Sequential Unidirectional dataflow applications graphs are generated. In each graph application category, 25% of interactive Real-Time (RT) applications, 25% of Streaming applications (ST), 25% of Mission Critical applications (MC) and 25% Best Effort applications (BE) are deployed. Each application has one sensor/actuator pair placed randomly on infrastructure IoT devices. Infrastructure sensors send simultaneously data item every 5ms. We stop simulation when all sensors have sent 500 data packets.

2) A fixed three layered Physical topology: According to [32], [33] we fix an average topology of three layers: Cloud layer, Proxy server, Gateways layer and IoT devices layer. Different access network technologies (4G LTE, WiFi, LPWAN, Wired) are used as real system. For data center and proxy server nodes, we use power information measured in a local data center composed of 4 physical nodes (Intel dual Xeon E5 2699 v3 18-cores). Processor specification

TABLE I: IoT applications classes and priority

QoS / Class	Best-Effort (BF)	Streaming (ST)	Real-Time (RT)	Mission Critical (MC)
Delay-sensitivity (ms)	–	150	50	20
Bandwidth demand	Low	High	High	High
Communication frequency	Low	Medium	High	High
CPU demand	Low	Low-Medium	Medium-High	Medium-High
Data Location	Remote	Local-Vicinity-Remote	Local-Vicinity-Remote	Local-Vicinity-Remote
Mobility	High-medium-low	High-medium-low	High-medium-low	High-medium-low
Examples	File sharing	Augmented reality games	video streaming	health tracking
Priority	3	2	1	0

TABLE II: Fixed Topology Nodes Features
$j \in [0, 3]$

L	Name	N	Devices	MIPS	RAM (MB)	uBW (Mbs)	P^{max}-P^{idle}
0	cloud	4	Cloud	120000	64000	10000	318-145
1	Fog	1	Proxy server	60000	8000	10000	169-70

2		2	d0(LoRa)
			d1(Wired)
				6750
				6750
					1000
					1000
						10000
						10000
							10-45
							10-45

2		2	d2(Wi-Fi)
			d3(4G-LTE)
				13500
				13500
					2000
					2000
						10000
						10000
							20-70
							20-70

3	IoT	16	m0-j m1-j m2-j m3-j	2800 4500 2800 4500	1000 3000 1000 3000	0.25/1/1000/1000 0.25/1/1000/1000 0.25/1/1000/1000 0.25/1/1000/1000	5.1-1.9 6.1-1.1 5.1-1.9 6.1-1.1

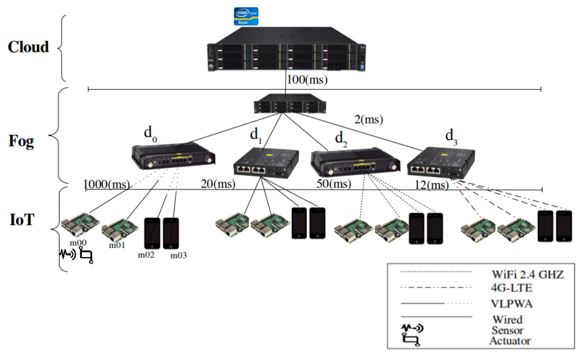
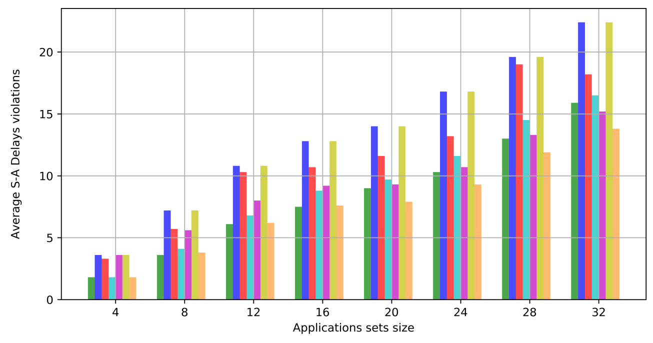


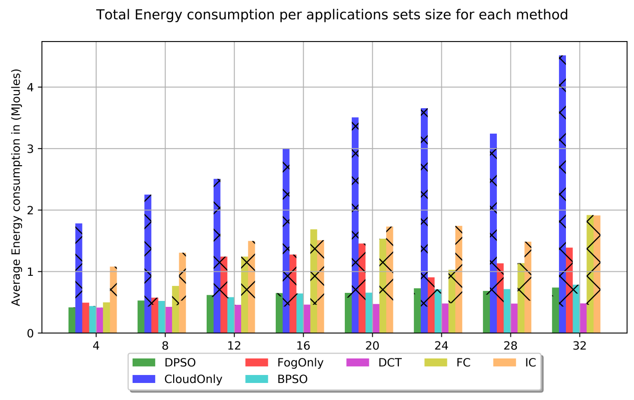
Fig. 4: The fixed topology used for the experimentation

comes from Intel web site and MIPS evaluation from 7-zip benchmark results [44], [46]. For Fog infrastructure routers, we took information of CISCO IR809 and IR829 [45]. For IoT device layer, we consider two types of equipments : an average capacity smartphone similar to Samsung galaxy S5 and a Raspberry pi 3+ B [47], [48]. Also, considering simulator constraints, the physical topology is a k-ary tree that represents a hierarchical multi-layer topology as detailed in Figure 4 and Table II. We use 4G-LTE and LoRa networks which are considered as attractive technologies for IoT environments. We also used more classical networks technologies (Wi-Fi and wired) highly deployed all over the world.

Considering 50 independent runs, Figure 5 shows both average number of delays violations (a) and system's energy consumption (b) obtained by each method for different size of heterogeneous applications sets, with hatched bar part representing network energy consumption and the non hatched one is for computation energy consumption. From 5, we can see that DPSO offers the best delay-Energy trade-off, followed by BPSO that is less efficient, which is due to

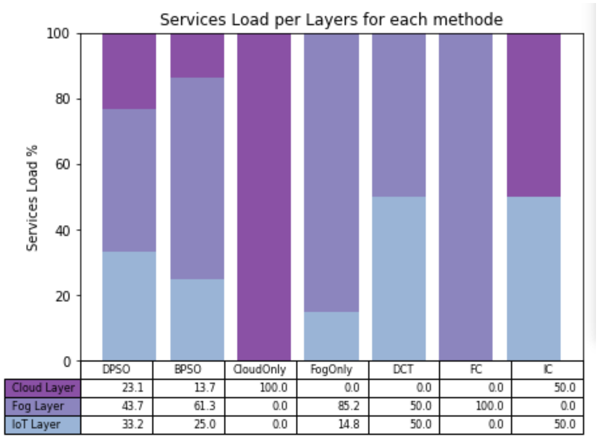


(a) Applications total delay violations



(b) System's energy consumption

Fig. 5: Total delay violations and Energy consumption for each placement method



	DPSO	BPSO	CloudOnly	FogOnly	DCT	FC	IC
Cloud Layer	23.1	13.7	100.0	0.0	0.0	0.0	50.0
Fog Layer	43.7	61.3	0.0	85.2	50.0	100.0	0.0
IoT Layer	33.2	25.0	0.0	14.8	50.0	0.0	50.0

Fig. 6: Services Load per each layer and for each placement method.

Sigmoid function saturation issues that leads at some points to reduce its exploration capacity. Then, DCT strategy is the third one. CloudOnly strategy is energy and network greedy. Using cloud for services that could be hosted on smaller devices can drastically increase energy consumption and response delay. In contrast, using only IoT and Fog layers like IoTFogOnly approach decreases network resources usage but services execution will take more time and induce performance degradation. Moreover, Fog and IoT layers are usually not sufficient to host all services computation, memory and storage needs (we stress that for our experiments nodes use time sharing overbooking scheduling strategy [26]. If the CPU request of a service is smaller than the CPU capacity of the node, allocation will be possible even if the total CPU requirement of all services on that node is higher than its capacity. Services will access the processor during a quantum in a time sharing way). IoTFogOnly strategy, by using only IoT and Fog layers, reduces energy consumption by reducing Network communications and uses less power greedy devices but in contrast the execution time increases and that leads to more delays violations when applications sets size increases, as we can notice it in (a). Taking more time for execution can also leads in some cases to consume more energy. Including cloud layer in the placement strategy is necessary.

With IC and FC approaches, which respectively uses only IoT-cloud and Fog-clod layers, we can notice that the first method is more energy greedy by adding cloud and network distance while it slows the overall computation capacities with powerless IoT nodes. In contrast, the approach is more interesting in term of delay by using IoT nodes proximity to place some services and exploiting cloud fast computation ability. FC strategy reduces energy consumption with smaller execution time while using more powerful nodes execution of service but in contrast it increases actuator response time. We deduce that using only two layers is not efficient to ensure a good delay-energy trade-off. DCT approach uses the 3 layers for the placement which gives good energy values but in contrast delay violations are higher than DPSO. The approach encourages placement in IoT and Fog layers and takes cloud layer in last choice and with the time sharing scheduling strategy it becomes almost similar to IoTFogOnly placement but still better than this last because it sorts respectively services and nodes by their increasing computation needs and capacities and so it uses fairly the exploited layers.

Figure 6 plots services load per each layer for each placement method. It confirms what was said previously and shows that DPSO makes a fair load balancing of services between all layers to deal with delay constraints and energy consumption minimization. DCT uses fairly IoT and Fog layer and does not use cloud layer because services can be placed on these nodes. Then, the scheduler will use time sharing between services. Even if simple heuristics are faster than evolutionary algorithms such as DPSO we can see that the QoS and energy gain are more interesting than other strategies.

DPSO takes for the placement of a 3 services application, with half of the time dedicated to simulation, 109s while CloudOnly, IoTFogOnly, IC, FC and DCT takes respectively 84ms, 211ms, 114ms, 241ms and 167ms. In contrepart, DPSO energy delays violations average gain are respectively 80% and 31% for CloudOnly, 38% and 31% for IoTFogOnly, 61% and 31% for FC, 38% and -15% for IC, 9% and 7% for BPSO and -30% and 15% for DCT.

From what was observed previously, we can say that DPSO exploits smartly both Fog and Cloud layers advantages respectively by reducing response delay violations and offering fast and powerful computation capacity while reducing as much as possible system energy consumption.

Conclusion

In this paper, we proposed a DPSO approach for IoT services placement in a physical Fog topology, we compared results with BPSO, CloudOnly and with different layers-interplay combination placement approaches through iFogSim. The main contribution of our proposal is in the conjunction of the following features: evaluation of total system energy consumption using a swarm intelligence based algorithm, defining applications' class with delay constraints, proposing an algorithm to map IoT services in a Fog infrastructure minimizing energy consumption. DPSO finds a good trade-off between cloud, Fog and IoT layers usage. Experimentation shows that DPSO prefers the middle layers devices, then depending on the application it chooses between IoT device and Cloud. For our future works, we plan to include network technologies specifics and applications priority in the placement policy logic. We will implement a multi-objective version of the DPSO with estimation strategies in order to improve computation time and solutions accuracy. We will also deal with a dynamic context by integrating objects mobility.

References

[1] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, "Fog Computing and Its Role in the Internet of Things", MCC 12 Helsinki Finland, 2012.

[2] M. Iorga, L. Feldman, R. Barton, M. Martin, N. Goren, C. Mahmoudi, ”Fog Computing Conceptual Model Recommendations of the National Institute of Standards and Technology”, NIST Special Publication 500-325, 2018.

[3] H. Gupta, A. Dastjerdi, S. Ghosh, R. Buyya, ”iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments”, IEEE Journal of Software : Practice and Experience , vol. 47, Special Issue: Cloud and Fog Computing, 2017.

[4] F. Jalali, S. Khodadustan, C. Gray, K. Hinton and F. Suits, ”Greening IoT with Fog: A Survey,” 2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI, 2017, pp. 25-31.

[5] Cisco company, Cisco Fog Computing with IOx,https://www.cisco.com/c/en/us/products/cloud-systems-management/iox/index.html, 2014.

[6] Cisco company, Linux Foundation project EdgeXFoundary, https://www.edgexfoundry.org, 2018.

[7] M. Taneja, A. Davy, ”Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm”, 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, 2017, pp. 1222-1228.

[8] L. Liu, Z. Chang, X. Guo, S. Mao and T. Ristaniemi, ”Multiobjective Optimization for Computation Offloading in Fog Computing”, IEEE Internet of Things Journal, vol. 5, no. 1, pp. 283-294, Feb. 2018

[9] J. C. Guevara, L. F. Bittencourt and N. L. S. da Fonseca, ”Class of service in fog computing,” 2017 IEEE 9th Latin-American Conference on Communications (LATINCOM), Guatemala City, 2017, pp. 1-6.

[10] N. K. Giang, M. Blackstock, R. Lea and V. C. M. Leung, ”Developing IoT applications in the Fog: A Distributed Dataflow approach,” 2015 5th International Conference on the Internet of Things (IOT), Seoul, 2015, pp. 155-162.

[11] IEA,”Energy Efficiency of the Internet of Things”,IEA 4E EDNA Technology and Energy Assessment Report,p. 1-66,2016.

[12] A. Brogi, S. Forti, ”QoS-Aware Deployment of IoT Applications Through the Fog”, IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1185-1192, Oct. 2017.

[13] O. SkarlatEmail M. Nardelli S. Schulte M. Borkowski P. Leitner,”Optimized IoT service placement in the fog”,Service Oriented Computing and Applications vol. 11,2017.

[14] S. Yi, C. Li, Q. Li, ”A Survey of Fog Computing: Concepts, Applications and Issues”, Mobidata 15, 2015.

[15] X. Meng, W. WangW, Z. Zhang,”Delay-Constrained Hybrid Computation Offloading With Cloud and Fog Computing”,IEEE Internet of Things Journal,vol. 5,2017.

[16] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu and Z. Han, ”Computing Resource Allocation in Three-Tier IoT Fog Networks: A Joint Optimization Approach Combining Stackelberg Game and Matching,” in IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1204-1215, Oct. 2017.

[17] Q. T. Minh, D. T. Nguyen, A. Van Le, H. D. Nguyen and A. Truong, ”Toward service placement on Fog computing landscape,” 2017 4th NAFOSTED Conference on Information and Computer Science, Hanoi, 2017, pp. 291-296.

[18] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana and M. Parashar, ”Mobility-Aware Application Scheduling in Fog Computing,” in IEEE Cloud Computing, vol. 4, no. 2, pp. 26-35, March-April 2017.

[19] Xuan-Qui Pham and Eui-Nam Huh, ”Towards task scheduling in a cloud-fog computing system,” 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), Kanazawa, 2016, pp. 1-4.

[20] R. Aparicio-Pardo and L. Sassatelli, ”A cost model for green fog computing and networking,” 2017 19th International Conference on Transparent Optical Networks (ICTON), Girona, 2017, pp. 1-5.

[21] A. Bozorgchenani, D. Tarchi and G. E. Corazza, ”An Energy-Aware Offloading Clustering Approach (EAOCA) in fog computing,” 2017 International Symposium on Wireless Communication Systems (ISWCS), Bologna, 2017, pp. 390-395.

[22] A. Mebrek, L. Merghem-Boulahia and M. Esseghir, ”Efficient green solution for a balanced energy consumption and delay in the IoT-Fog-Cloud computing,” 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, 2017, pp. 1-4.

[23] C. Sonmez, A. Ozgovde and C. Ersoy, ”EdgeCloudSim: An environment for performance evaluation of Edge Computing systems,” 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), Valencia, 2017, pp. 39-44.

[24] C. Gray, R. Ayre, K. Hinton, R. S. Tucker, ”Power consumption of IoT access network technologies,” 2015 IEEE International Conference on Communication Workshop (ICCW), London, 2015, pp. 2818-2823.

[25] D. Evans, ”The Internet of Things How the Next Evolution of the Internet Is Changing Everything”,Cisco Internet Business Solutions Group (IBSG),2011.

[26] L. Toms, J. Tordsson, ”Improving Cloud Infrastructure Utilization through Overbooking”, ACM International Conference on Cloud and Autonomic Computing, 2013.

[27] https://www.planetoscope.com/electronique/230-energie-consommee-par-les-data-centers.html, January 2018.

[28] M. R. Mahmud R.Buyya, ”Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit”,2018.

[29] J. C. Guevara, L.F.Bittencourt, J. Diaz-Montes,R. Buyaa,O. F. Rana, M. Parashar,”Multiple Workflows Scheduling in Multi-tenant Distributed Systems: A taxonomy and Future Directions”,ACM Computing Surveys vol. 1,2018.

[30] N. Abbas, Y. Zhang, A. Taherkordi and T. Skeie, ”Mobile Edge Computing: A Survey,” in IEEE Internet of Things Journal, vol. 5, no. 1, pp. 450-465, Feb. 2018.

[31] N. Fernando, S. W. Loke, W. Rahayu,”Mobile cloud computing: A survey”,ELSEVIER Future Generation Computer Systems,Vol. 29, Issue 1,pp 84-106,2013.

[32] P. Wang, S. Liu, F. Ye, X. Chen,”A Fog-based Architecture and Programming Model for IoT Applications in the Smart Grid” vol. 1,2018.

[33] H. Gupta, Dr. Ajay K. Bharti,”Fog Computing IoT: Overview, Architecture and Applications”,International Journal of Advanced Research in Computer and Communication Engineering vol. 1, 2018.

[34] R. N. Calheiros R.Ranjan C.A.F.De Rose R.cBuyya,”CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services”,ICPP,2009.

[35] R. C. Eberhart and Y. Shi, ”Comparing inertia weights and constriction factors in particle swarm optimization,” Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), La Jolla, CA, USA, 2000, pp. 84-88 vol.1.

[36] J. Kennedy and R. C. Eberhart, ”A discrete binary version of the particle swarm algorithm,” 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, USA, 1997, pp. 4104-4108 vol.5.

[37] F. Marini, B. Walczak, ”Particle Swarm opritmization (PSO).A tutorial”,Chemometrics and intelligent Laboratory Systems ELSEVIER,2015.

[38] J. Kennedy, R. C. Eberhart,”Particle Swarm opritmization”,Proceedings of the IEEE International Conference on Neuronal Networks vol. 4,1995.

[39] J. Xin G. Chen Y. Hai,”A particle swarm optimizer with multistage linearly-decreasing inertia weight”,Proceedings of the International Joint Conference on Computational Sciences and Optimization,vol. 1,2009.

[40] A. Salman, I. Ahmad, S. Al-Madani,”Particle swarm optimization for task assignment problem”,Microprocessors and Microsystems vol. 26, 2002.

[41] A.E. Muoz Zavala,”A comparison study of PSO neighborhoods”, Springer pp. 251265, Heidelberg, 2013.

[42] H. Izakian B.T. Ladani A. Abraham ,V. Snel,”A Discrete particle swarm optimization approach for grid j ob scheduling”,International Journal of Innovative Computing, Information and Control vol. 6, 2010.

[43] A. Singh S. Rathkanthiwar S.Kakde,”Energy efficient routing of WSN using particle swarm optimization and V-LEACH protocol”, International Conference on Communication and Signal Processing (ICCSP), 2016.

[44] Intel, https://ark.intel.com/products/84682/Intel-Xeon-Processor-E7-8870-v3-45M-Cache-2-10-GHz, December 2018.

[45] Cisco iox, https://community.cisco.com/t5/cisco-iox-documents/cisco-iox-nodes-isr819-cgr1120-1240-ir829-809-hardware-and/ta-p/3619076

[46] CPU benchmark, https://www.7-cpu.com/, December 2018.

[47] Raspberry benchmark,https://medium.com/@ghalfacree/benchmarking-the-raspberry-pi-3-b-plus-44122cf3d806, December 2018.

[48] Samsung Galaxy S5 Review, https://www.notebookcheck.net/Review-Samsung-Galaxy-S5-Smartphone.116068.0.html, December 2018.