Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8899660, 12 pages
https://doi.org/10.1155/2021/8899660

WILEY | Hindawi

Research Article

New Heuristics for Scheduling and Distributing Jobs under Hybrid Dew Computing Environments

Pablo Sanabria,[1] **Tomás Felipe Tapia**,[1] **Andres Neyem**,[1] **Jose Ignacio Benedetto**,[1] **Matías Hirsch**,[2] **Cristian Mateos**,[2] **and Alejandro Zunino**[2]

[1]*Department of Computer Science, Pontificia Universidad Católica de Chile, Santiago, Chile*
[2]*ISISTAN Research Institute, CONICET-UNICEN, Buenos Aires, Argentina*

Correspondence should be addressed to Pablo Sanabria; psanabria@uc.cl

Received 14 August 2020; Revised 1 February 2021; Accepted 11 February 2021; Published 4 March 2021

Academic Editor: Rahul Yadav

Mobile grid computing has been a popular topic for researchers due to mobile and IoT devices' ubiquity and their evergrowing processing potential. While many scheduling algorithms for harnessing these resources exist in the literature for standard grid computing scenarios, surprisingly, there is little insight into this matter in the context of hybrid-powered computing resources, typically found in Dew and Edge computing environments. This paper proposes new algorithms aware of devices' power source for scheduling tasks in hybrid environments, i.e., where the battery- and non-battery-powered devices cooperate. We simulated hybrid Dew/Edge environments by extending DewSim, a simulator that models battery-driven devices' battery behavior using battery traces profiled from real mobile devices. We compared the throughput and job completion achieved by algorithms proposed in this paper using as a baseline a previously developed algorithm that considers computing resources but only from battery-dependent devices called Enhanced Simple Energy-Aware Schedule (E-SEAS). The obtained results in the simulation reveal that our proposed algorithms can obtain up to a 90% increment in overall throughput and around 95% of completed jobs in hybrid environments compared to E-SEAS. Finally, we show that incorporating these characteristics gives more awareness of the type of resources present and can enable the algorithms to manage resources more efficiently in more hybrid environments than other algorithms found in the literature.

1. Introduction

The popularity of mobile devices and their integration with the Internet of Things in different environments have increased the need to improve network technologies and mobile devices' capabilities. Cisco estimates that we will have around 12.3 billion mobile devices connected around 2022 [1]. In recent years, the hardware in mobile devices has been improved following Moore's law. These days, we can find cheaper devices twice as powerful in regard to their computation, memory, and storage than devices present two years ago, capable of executing and managing complex tasks [2]. Due to these advances in mobile computing, researchers [3] found the need to treat these mobile devices as first-class resources in distributed computing environments, using these devices for complex tasks like executing Artificial Intelligence (AI) algorithms (for example, object detection, object tracking, and image recognition). Thus, it has become possible to integrate various mobile devices, such as smartphones, robots, sensors, and single-board computers (SBCs) in Cloud-Fog-Dew environments [4].

One of the most common reasons to connect mobile devices into a Cloud computing system is to increase mobile devices' capabilities using code offloading techniques, sending work to compute into high-end devices located in the Cloud [5, 6]. Using these techniques, developers can reduce battery consumption and increase mobile devices' capabilities further than their hardware specifications [7–9]. With the increasing demand for IoT devices in recent years [10], the Cloud has begun to suffer bottlenecks in throughput speed, latency, and storage. In response to this demand, researchers have proposed new network architectures to solve this problem. Concepts like Edge computing (encompassing Fog computing and Dew computing) bring

computation power to close devices located nearby in geographical space. While these two paradigms help reduce the network's problems, they still depend (especially Fog computing) on the network backbone that may not be available or reachable in certain situations like working with IoT devices in mines and on ships, in deserts, or in moving vehicles. Dew computing is a new paradigm where connected devices offload jobs to nearby devices in the same network. This paradigm proposes an architecture that tries to reduce network latency, the energy cost of remote data communication, and the costs inherent to Cloud infrastructure usage [11]. Through this, Dew computing tries to optimize the usage of mobile and IoT devices in the system. Firstly, it treats mobile devices as clients on the network infrastructure to offload their work to other devices located in the local network to increase their capabilities [12]. Secondly, Dew computing considers mobile and IoT devices and resources to increase the available computational power from an existing system. In this approach, one device can offload its work to another available device in the network (including other mobile and IoT devices) [13, 14].

A network topology is needed to use mobile and IoT as resources and to allow networking support to acquire resource availability knowledge, distribute tasks, and collect results [15]. The Smart Cluster at the Edge (SCE) is a topology for infrastructure-based networks and can be used in Edge and Dew computing environments. This topology could be established wherever an access point and a group of mobile and IoT devices coexist. This topology's main feature is that it has a central scheduler primarily used to coordinate the task assignment for the network's available resources and manage the different capabilities and characteristics of available resources connected to the network. This central scheduler can be a server in Edge environments and any other capable device in Edge and Dew environments [16]. To use these resources, we need to consider these devices' particular traits like the power source (and the battery capacity for battery-dependent devices) and the number of processors/CPU cores, storage, or sensors. These features, along with the set of assigned jobs and information on available resources, result in many challenges for SCE [17, 18]. Several scheduling algorithms for CPU-bound jobs in SCE are available in the current literature [15, 19]. However, these algorithms only consider mobile devices with limited power supply and disregard nodes hooked to a power grid which lack this constraint.

The contributions of this work are as follows:

(i) Improve existing resource allocation with three new algorithms: Batch Processing Algorithm, Weighted Random, and Weighted Round Robin. Those new algorithms were designed for exploiting clusters of Edge and Dew nodes with different computing capabilities provided by the battery- and non-battery-dependent devices. Examples of battery dependent devices are smartphones, tablets, smartwatches, and portable IoT devices; for non-battery-dependent devices, we consider SBCs (single-board computers) or other IoT devices directly connected to a power grid

(ii) Create an enhanced version of DewSim [3] to support hybrid networks to test our proposed algorithms

The rest of the paper is organized as follows. In the next section, we will discuss state-of-the-art scheduling algorithms used in SCE environments. Section 3 will show our contribution to an existing simulator called DewSim to support mixed SCE environments. Section 4 will show our proposed scheduling algorithms for SCE. Section 5 will show our methodology and results from our experiments in the extended simulator using our proposed heuristics. In Section 6, we will show our conclusions about the obtained results and the planned future work.

2. Related Work

2.1. Scheduling Algorithms for Dew Computing. Exploiting the computing capabilities of SCEs is a complex task that requires scheduling logic to efficiently use smart devices, such as smartphones and tablets, as a special kind of provider node which is computing the available energy conditions' service provision in their primary source of power (batteries). Several scheduling algorithms have been proposed regarding this and other aspects concerning resource scavenging using this type of smart device. For instance, in [20], the proposed scheduling algorithm considers performance indicators of wireless network bandwidth and mobile helper devices' presence time to schedule jobs of a workflow with different data dependencies. In [21], the task scheduling algorithm is modeled as a mixed-integer programming problem that considers unstable network links due to node mobility and resources placed outside the local context where tasks are originated. In [22–26], different algorithms were proposed for maximizing system utility or minimizing job execution time, using mobile devices' remaining energy as a formal constraint of the resource allocation problem formulation. Such data is easy to estimate by accessing the battery level through a smartphone battery API. However, to operate, these algorithms also assume complete and accurate information about job requirements in terms of energy spent and execution time for every candidate node, making it challenging to apply in real-life scenarios.

In [27, 28], another kind of scheduler algorithm is proposed that does not need complete knowledge of job requirements to operate. In other words, they do not need the job information that algorithms discussed above require, e.g., job execution time or job energy spent in each candidate node. These algorithms focus on reducing the job completion time by exploiting nodes' proximity and cost-effectiveness of nodes' transferring capabilities. Their primary focus is to study node mobility's effect on the job completion time rather than balancing the load among nodes to mitigate the limited operation time imposed by their batteries and their computing resources.

Ranking-based heuristics from Hirsch et al. [16, 17] combine mobile devices' battery level information (remaining energy or discharge time) with computing scores resulting from benchmarks to exploit a set of nearby smartphones' computing capabilities. The authors show that considering

nodes' remaining energy helps to complete more jobs with a set of battery-powered mobile devices than using a classical Round Robin scheduler. However, such heuristics assume that the burden of processing all jobs is exclusively in charge of battery-powered mobile devices. That means that such heuristics' performance is unknown for settings where the battery- and non-battery-powered devices coexist in the same Dew computing environment. Besides, non-battery-powered hardware is severely underutilized.

The last scenario could be numerous local smart contexts, where SBCs and similar IoT devices support much of the processing burden. However, the extra load derived from fortuitous events or fluctuating demands is not well supported without additional resources, such as passive battery-powered devices in proximity. For exploiting such settings, new load-balancing heuristics should be investigated, which is the main objective of this work.

In general, these existing algorithms show that they can improve the performance and job completion rate of SCEs. However, they are unable to handle cases where non-battery-powered devices are part of the network. If we consider these resources, we must address new requirements because there is a need to distribute the jobs in an optimal way considering these two types of devices: battery- and non-battery-dependent devices.

2.2. Dew Computing Simulation. Simulation is an accepted practice in distributed computing because it allows researchers to simulate events and heuristics without the cost of hardware and infrastructure, with the consequent reduction in evaluation times. Also, simulation provides a way to repeat experiments easily without worrying about costs, and it eases the process of simulating real-life events.

Simulation frameworks with built-in support for modeling computing-related concerns are commonly used for grid computing research. One notable example is GridSim [29]. This event-driven simulation toolkit provides abstractions for modeling large-scale distributed computing systems in which millions of resources with single or multiprocessors, with shared or distributed memory managed by time- or space-shared schedulers, are integrated.

SimGrid [6] is a more versatile option, a tool with specific models for simulating various distributed systems, including clusters, content sharing in extensive and local area networks, data centers, and Cloud environments. However, these models only allow simulating the behavior of dedicated computing resources connected through wired networks. Modeling nondedicated computing resources is a crucial aspect of mobile grid computing.

CloudSim [30] is a toolkit for simulating Cloud computing infrastructures. It provides different abstractions to represent virtual machines running in a server collection located in a data center. It is used to study different ways to manage Cloud computing scenarios.

EdgeCloudSim [31] provides a simulation environment specific to Edge computing scenarios where it is possible to conduct experiments that consider both the computational and networking resources. EdgeCloudSim is based on CloudSim but adds additional functionalities to be used for Edge computing scenarios more efficiently. However, this simulator has the same problem as CloudSim, and it cannot simulate interactions and resource sharing between mobile devices. Similarly, Flores et al. [32] built a simulator based on CloudSim that adds mobile devices' presence to allow task offloading to the Cloud. It also adds the concept of battery-dependent devices but cannot simulate interactions with other mobile devices connected to the network.

iFogSim [33] permits model entities present in IoT scenarios. It uses CloudSim as a base simulation engine. It provides abstractions to model sensors and actuators, commonly associated with the data source and data sink roles, and Fog nodes that can be loaded with different modules to serve different IoT applications. Fog nodes play the role of computing and/or data relays to communicate with other Fog nodes or even the Cloud to offload computations. Fog nodes can be parametrized with power consumption values to enable studies that consider energy-aware resource allocation. However, to model battery-dependent Fog nodes, such as smartphones and tablets, it is essential to consider battery behavior and not only power consumption.

IoTSim-Edge [34] is another framework built upon CloudSim. In this simulator, low-powered devices such as smartphones and Raspberry Pi are considered first-class computing resource providers, i.e., Edge data centers' Edge nodes. The framework considers modeling battery drainage of Edge nodes and IoT devices through linear relationships between resource usage and battery capacity which is known to be an oversimplified way of representing the way batteries behave.

Smartphones are also modeled as first-class computing resource providers in DewSim [3]. In DewSim, a trace-driven approach is used to model battery behavior. It means that battery level decrements are obtained from real battery traces profiled from mobile devices, representing realistic relations between computing resource utilization and battery drainage. DewSim is considered an infrastructure-level topology where mobile devices are registered with a proxy node as resource providers. The proxy is responsible for assigning received jobs to mobile devices. The simulator has been used with allocation algorithms where jobs are distributed only among battery-dependent devices. Resource allocation logic to exploit the synergy between the battery- and non-battery-dependent devices has not been explored yet. On the other hand, this simulator has an architecture that allows adding new types of nodes and scheduling algorithms using its interfaces.

Table 1 summarizes the main characteristics of these frameworks and their capabilities. It shows existing simulators in the literature that facilitate the research on resource allocation in distributed computing; the first column shows the type of nodes on which the distributed computing is performed. The next column shows if the simulator supports mobile devices as a client or as a computing device. The third column shows if the simulation allows battery modeling for mobile devices and which type of modeling is used. The offloading task column indicates if they support task offloading simulation to other network devices and where the task can be offloaded. The "Energy discharge profiles" field shows if

Table 1: Distributed computing simulator comparison table.

Simulator	Computing node	Mobile device role	Battery modeling	Task offloading	Energy discharge profiles
GridSim	High-end servers in a cluster	N/A	N/A	Between dedicated servers	No
SimGrid	High-end servers in a cluster	N/A	N/A	Between dedicated servers	No
CloudSim	High-end servers located at the Cloud	Client	N/A	Cloud devices and dedicated servers	No
EdgeCloudSim	Mid/high-end servers located at the Cloud and the Edge	Client	No	Cloud devices and dedicated servers	No
MobileCloudSim	High-end servers located at the Cloud	Client	No	Cloud devices and dedicated servers	No
iFogSim	Mid-end servers located at the Fog	Client	No	Devices located at the Fog	No
IoTSim-Edge	Mobile devices	Computing device	Linear model	Mobile devices	No
DewSim	Mobile devices	Computing device	Trace-driven	Mobile devices	Yes

the simulation supports real devices' use and discharge rate in different energy resource demands (e.g., screen on/off, CPU loads, and network usage).

3. DewSim Extended Simulator

Since the simulators available in the literature were not prepared to handle topologies based on mixed networks or capable of handling battery- and non-battery-dependent devices, we built a custom simulator (available code and simulated environments in https://github.com/psanabriaUC/mobile-grid-simulator) to adequately address these scenarios. We based our simulator on the previous work of DewSim [3]. DewSim was chosen as a base simulator because it supports modeling different features present in an SCE. The main features of DewSim are the simulation of the arrival of tasks, completeness metrics, battery consumption, network activity derived from the transfer of input/output data of tasks, and status notifications of devices based on events. The simulator allows modeling battery consumption in mobile devices thanks to its method based on profiles extracted from real devices, that is, traces that contain information (not synthetic) about the relationship between battery events and CPU usage.

DewSim manages all the devices using the device class, as shown in the class diagram in Figure 1. The device class has all inherent attributes that can describe a resource in the network. This device interacts with two essential classes: the battery manager, which manages information related to the remaining energy available in the devices, and the execution manager, which manages information related to the computation of jobs assigned to a device.

DewSim handles the device's battery depletion simulating different events that cause energy depletion and estimates in this way, by using the battery profiles extracted from real devices, the remaining battery. The simulated events supported by DewSim are events related to CPU usage, network usage, or screen activity. DewSim initially did not handle devices with unlimited power supply. We added a new device extending a new battery manager that implements the same interface implemented before but adapted to an infinite battery source in our new version. Because the energy source is not limited to a battery and it will always have a constant energy source, the new logic implemented in this battery manager is that it always reports 100% of the remaining battery to the scheduler and does not respond to energy depletion events.

We added a new attribute to the devices present in the network. This attribute is used to distinguish between both types of devices. It helps scheduling algorithms and the event manager discriminate between devices and decides how they can handle the arrived jobs to the SCE.

Figure 2 shows the logic behind the behavior of the simulator. When the simulation begins, it receives a list of events, like job arrivals, battery status updates, and message sending between devices. The simulation ends when all the events are processed. Every event is sent to a device to process it and send the job information to the battery manager. The battery manager is responsible for calculating the remaining battery using the job information and sending the corresponding event to be processed in the simulation. In non-battery devices, the default infinite battery manager class maintains the behavior of the base battery manager. However, it always sends its battery status as a fully charged device. In this way, the simulation maintains its consistency and can manage both the battery and non-battery devices using the same interfaces.

4. Proposed Scheduling Heuristics

There are various scheduling algorithms for SCE systems. To analyze the consequences of adding non-battery-dependent mobile devices, we need to test them when adding this type of device into SCE systems. In this work, we will test current algorithms designed to be used with mobile grid systems. Rodriguez et al. [35] proposed the SEAS (Simple Energy-Aware Schedule) algorithm to minimize the per-device energy consumed per job executed. This algorithm's main feature is that it is easy to implement in real-life environments, using only OS (Operative System) information

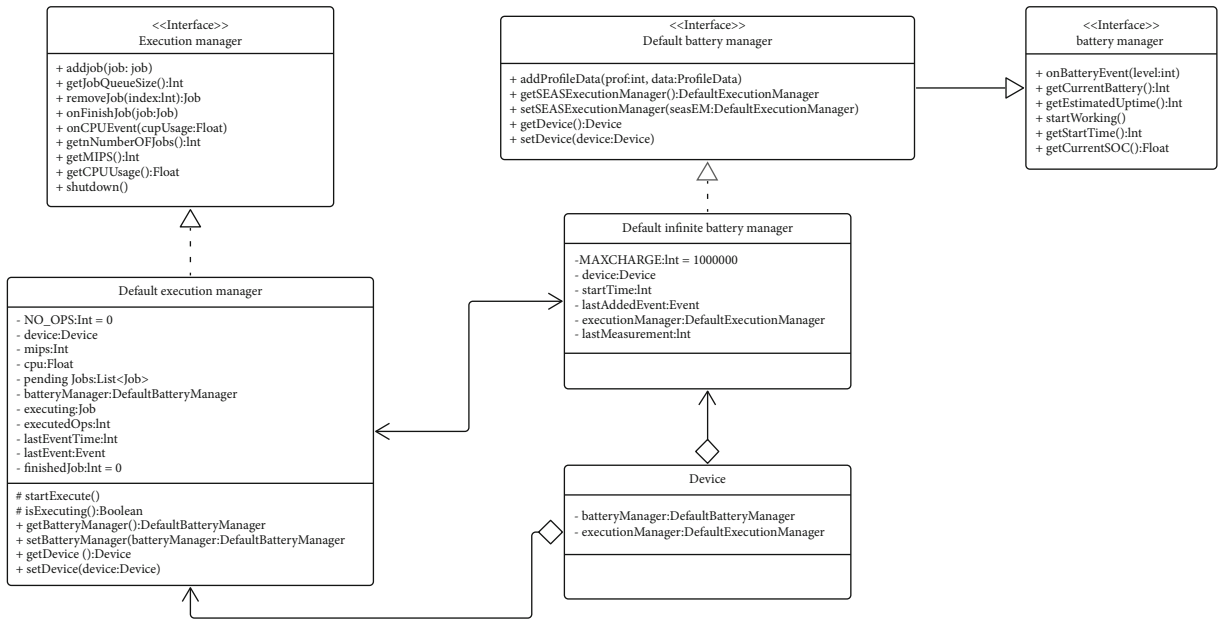

Figure 1: Class diagram of the infinite battery manager.

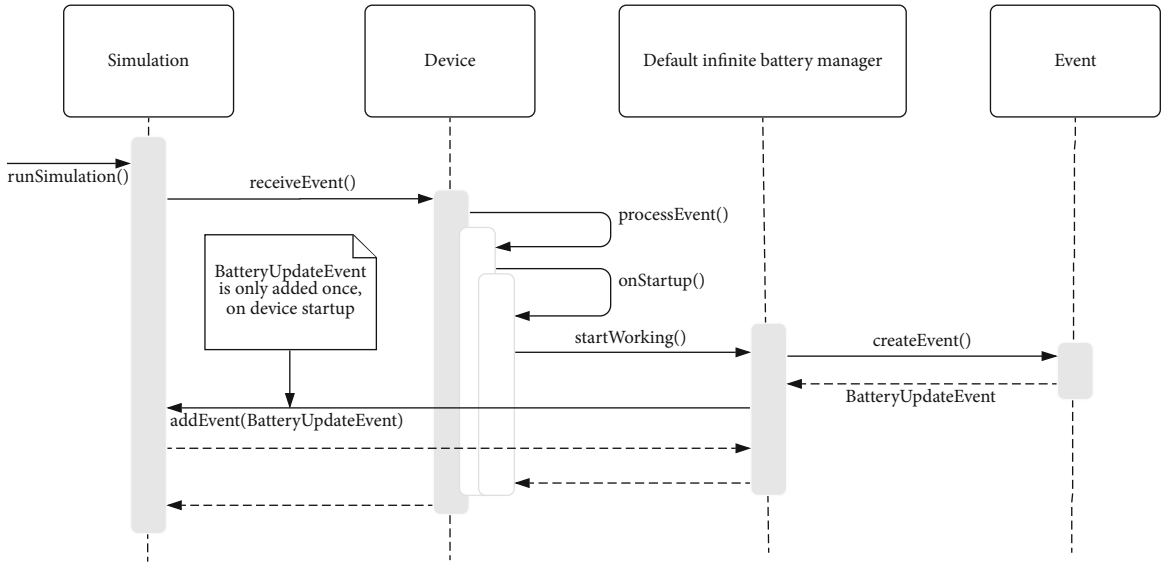

Figure 2: Sequence diagram of the infinite battery manager.

available in conventional power-constrained devices. This algorithm operates by assigning jobs to a centralized proxy that manages and distributes the jobs to the network's available resources. It ranks all subordinated mobile devices according to which one can be assigned a job and knows which device is the best candidate to do the job. It uses the estimated device uptime, the device's capabilities (previously measured with benchmarks), and the remaining jobs. Hirsch et al. [17] extended this algorithm. They called it Enhanced-SEAS. This algorithm uses the combination of three components, including the current node battery level, the computation capabilities of the node measured in MIPS, and the jobs the node has in its job queue.

Additionally, other relevant algorithms are called Job Energy-aware Criterion (JEC) and Future Work-aware Criterion (FWC). The first algorithm considers the relationship between the energy used and the number of jobs finished by a device. FWC, on the other hand, considers that the future computational power of a node could be estimated by analyzing the computational power the node presented in the past. In other words, FWC assumes that the throughput achieved by a node in the past could be maintained in the future as well [17].

This paper proposes three new scheduling algorithms inspired by algorithms such as Batch Processing Algorithm, Weighted Random, and Weighted Round Robin. The Batch

```
1 : assignJob (job)
2 :     assignedOPS = deviceList . getCurrentLoad ()
3 :     selected   ← null
4 :      foreach device in deviceList
5 :             if (selected == null)
6 :                 selected   ← device
7 :             else
8 :                 dt_selected   ← (job . OPS + assignedOPS [selected]) /
9 :                                 selected . FLOPS ∗ selected . batteryLevel
10 :                dt_device     ← (job . OPS + assignedOPS [device]) /
11 :                                device . FLOPS ∗ device . batteryLevel
12 :                 if (dt_device < dt_selected)
13 :                     selected ← device
14 :                end
15 :            end
16 :    end
17 :     if (selected != null)
18 :         queueJob (selected, job)
19 :    end
20 : end
```

Figure 3: Pseudocode of the Batch Processing Algorithm scheduler.

Table 2: Execution environments used in experiments.

Device	MFLOPS	Total battery (joules)	Battery-dependent only	Hybrid	Non-battery-dependent only
A100	61.665	40680	20	10	0
Galaxy Tab 2	179.832	53280	30	15	0
L9	56.433	29520	50	25	0
Raspberry Pi 3	58.380	N/A	0	35	70
ODROID XU4	35.770	N/A	0	35	70

Processing Algorithm (BPA) considers the job loaded into the devices and the current hardware capabilities to predict the first device that finishes the assigned job. The scheduler assigns and enqueues the job to the predicted device. The formula we use for this job is as follows:

$$\Delta t = \frac{\sum \mathrm{OP}_{\mathrm{jobs}}}{\mathrm{MIPS} * \mathrm{Battery}}, \quad (1)$$

where $\mathrm{OP}_{\mathrm{jobs}}$ is the current job load that the device has enqueued measured in how many operations are needed to finish the task, MIPS is the device's capabilities (processor speed) measured in mega instructions per second, and Battery is the remaining battery of the device expressed in values between the range of $0 < \mathrm{Battery} \leq 1$. The pseudocode of this algorithm is in Figure 3, where "assignedOPS" is a hashtable containing a registry of the current load of the devices, "selected" is the device to be assigned a job, and "deviceList" is the list of devices connected to the network. We also apply a Job-Stealing technique, which means a device does not wait to be assigned jobs to execute them but asks other devices for unfinished jobs upon becoming idle. However, in this case, we treat the battery- and non-battery-dependent devices in different ways. We have two approaches. The first one is to assign the jobs using the same techniques proposed before, using our schedulers and the previously designed schedulers like E-SEAS, JEC, or FWC. The second approach is treating the non-battery-dependent devices as first-class devices. That means that the scheduler assigns jobs, using a previously studied algorithm, to non-battery-dependent devices first. After that, the scheduler waits for battery-dependent devices to steal jobs from the non-battery-dependent devices.

The Weighted Random (WR) algorithm works by assigning a weight to each device on the grid. The weight is simply a combination of the device's capability measured in MIPS multiplied by the current battery level (measured in values between 0 and 1). Therefore, the scheduler assigns a given job at random, but given the weight, devices with higher capabilities and more battery levels have a higher chance of being selected. For this study, we assigned the next range. For devices with more than 1000 MIPS, 20 jobs are assigned; for devices with more than 75 MIPS but less than 100 MIPS, ten jobs are assigned; for devices with more than 10 MIPS but less than 75 MIPS, seven jobs are assigned; and finally, for devices with less than 10 MIPS, one job is assigned.

The Weighted Round Robin (WRR) algorithm works using the classic Round Robin algorithm, assigning a previously determined number of jobs to each grid device until every job has been assigned. Again, this number depends on each device's capabilities multiplied by the battery level (measured in values between 0 and 1). For this study, we determined the following thresholds: devices with less than

Table 3: Obtained results for 1500 jobs.

Environment		GIPS	Completed jobs	Completed on non-battery	Completed on battery
Battery	Enhanced-SEAS	0.585115579	63.47%	0.00%	63.47%
	BPA	0.570180123	55.93%	0.00%	55.93%
	BPA-Stealing	0	0.00%	0.00%	0.00%
	W-Random	0.5677372	68.29%	0.00%	68.29%
	W-Round Robin	0.621309887	68.40%	0.00%	68.40%
Non-battery	Enhanced-SEAS	0.534927378	100.00%	100.00%	0.00%
	BPA	0.549990732	100.00%	100.00%	0.00%
	BPA-Stealing	0.035535733	100.00%	100.00%	0.00%
	W-Random	0.606253274	100.00%	100.00%	0.00%
	W-Round Robin	0.483890686	100.00%	100.00%	0.00%
Hybrid	Enhanced-SEAS	0.523443085	100.00%	100.00%	0.00%
	BPA	0.533033243	91.80%	52.80%	39.00%
	BPA-Stealing	0.035535733	97.07%	78.47%	18.60%
	W-Random	0.805195744	90.45%	49.96%	40.49%
	W-Round Robin	1.0406022	96.02%	55.53%	40.49%

10 MIPS are assigned one job; devices with more than 10 MIPS and less than 75 MIPS are assigned seven jobs; devices with more than 75 MIPS and less than 100 MIPS are assigned ten jobs; and devices with more than 100 MIPS are assigned 20 jobs. We defined some thresholds considering the simulated devices' capabilities, creating a heterogeneous distribution of work. In practice, assigned jobs could be determined similarly using a function that takes as input the number of available jobs and the spectrum of MIPS among devices in the SCE.

5. Experimentation

To know how well the scheduling algorithms manage the available resources in the network, we need to test and compare the performance and reliability of job completion and compare it against other existing algorithms. In this paper, we compare our new algorithms against E-SEAS. In previous works, E-SEAS has shown that it gets better results in managing systems with battery-dependent devices compared to other algorithms found in the literature [16]. As this is the case, we use E-SEAS as a baseline because it can hint at how our algorithms behave in similar situations. On the other hand, there is no evidence about this algorithm's, nor other previously developed algorithms', behavior in more hybrid networks likely found in Dew computing grids. For that reason, we also use E-SEAS as a baseline to hint at how other algorithms behave in hybrid networks and compare them with our algorithms.

5.1. Methodology. To compare the algorithms' performance, as mentioned earlier, we considered three distinct scenarios:

(1) Battery-Dependent Only. We only use a set of devices that are battery-dependent.

(2) Non-Battery-Dependent Only. We only use a set of devices that are non-battery-dependent.

(3) Mixed Setup. We use the same devices used in the previous two experiments.

The metric used in these experiments will be the overall system throughput in GIPS (giga instructions per second) and total job completion. For that reason, we will measure the total time used by the SCE and the total executed instructions in the simulator to get the GIPS. The obtained GIPS by the SCE tells us how efficient the system is at finishing the assigned tasks after the scheduling algorithms distribute the tasks: more GIPS would mean more efficiency finishing a given set of tasks. On the other hand, the number of finished jobs, measured as a percentage, will show us how a scheduling algorithm can handle the risk of losing jobs when a specific device becomes unavailable (more completed jobs would mean more awareness of the network).

5.2. Experimental Setup and Results. The topology used in different environments is explained in Table 2. We used a tool based on LINPACK benchmarks (https://play.google.com/store/apps/details?id=com.sqi.linpackbenchmark) to get the device processing capacity (MIPS). To get the CPU battery consumption profiles, we used a tool developed together with DewSim to get those profiles. The devices used to get a profile were as follows: Acer A100 tablet, Samsung Galaxy Tab 2 tablets, LG L9 smartphones, Raspberry Pi 3 Model B, and ODROID XU4.

The job datasets are composed of synthetic jobs whose input and output vary in size (between 1 and 500 MB) and the number of operations required to complete. CPU operations relate in $n * \log(n)$, n^2, or n^3 to the input data size in KB to generically express various real-life codes (e.g., sorting data or performing 2D/3D matrix operations). We used three job sets consisting of 1500, 2500, and 3500 jobs for each experiment in every topology. The obtained results for 1500 jobs are shown in Table 3, results for 2500 jobs are shown

Table 4: Obtained results for 2500 jobs.

Environment		GIPS	Completed jobs	Completed on non-battery	Completed on battery
Battery	Enhanced-SEAS	0.702226488	39.12%	0.00%	39.12%
	BPA	0.734560767	39.52%	0.00%	39.52%
	BPA-Stealing	0	0.00%	0.00%	0.00%
	W-Random	0.650641616	48.44%	0.00%	48.44%
	W-Round Robin	0.675704095	49.68%	0.00%	49.68%
Non-battery	Enhanced-SEAS	1.417140659	100.00%	100.00%	0.00%
	BPA	1.060698678	100.00%	100.00%	0.00%
	BPA-Stealing	0.035566736	100.00%	100.00%	0.00%
	W-Random	0.90438383	100.00%	100.00%	0.00%
	W-Round Robin	1.107414652	100.00%	100.00%	0.00%
Hybrid	Enhanced-SEAS	0.844525553	100.00%	100.00%	0.00%
	BPA	0.997901846	83.96%	52.28%	31.68%
	BPA-Stealing	0.038224314	98.32%	86.24%	12.08%
	W-Random	1.103836675	81.35%	50.25%	31.10%
	W-Round Robin	1.278510946	83.84%	55.48%	28.36%

Table 5: Obtained results for 3500 jobs.

Environment		GIPS	Completed jobs	Completed on non-battery	Completed on battery
Battery	Enhanced-SEAS	0.735704316	22.97%	0.00%	22.97%
	BPA	0.793065449	24.11%	0.00%	24.11%
	BPA-Stealing	0	0.00%	0.00%	0.00%
	W-Random	0.704779595	32.61%	0.00%	32.61%
	W-Round Robin	0.77958216	33.42%	0.00%	33.42%
Non-battery	Enhanced-SEAS	1.421876509	100.00%	100.00%	0.00%
	BPA	1.449449702	100.00%	100.00%	0.00%
	BPA-Stealing	0.03556378	100.00%	100.00%	0.00%
	W-Random	1.184283041	100.00%	100.00%	0.00%
	W-Round Robin	0.811127206	100.00%	100.00%	0.00%
Hybrid	Enhanced-SEAS	0.968977662	100.00%	100.00%	0.00%
	BPA	0.994339283	73.71%	52.71%	21.00%
	BPA-Stealing	0.034761667	98.46%	88.83%	9.63%
	W-Random	0.950427366	70.60%	49.68%	20.92%
	W-Round Robin	0.916302079	74.91%	55.40%	19.51%

in Table 4, and finally, results for 3500 jobs are shown in Table 5.

5.3. Discussion. Figure 4 shows the overall performance of the system. We can see that BPA and E-SEAS behave similarly in different environments and job loads. We observe that BPA has a range of 2% and 18% throughput gain over E-SEAS in non-battery and hybrid environments. On the other hand, we can see that WR and WRR have better throughput in hybrid environment situations than the other algorithms. In the case of WRR, this algorithm obtained 51% and 90% of throughput gain in hybrid environments when the job load was about 2500 and 1500 assigned jobs, respectively. On the other hand, the behavior was similar to E-SEAS when the job load was 3500 assigned jobs. WRR behavior was similar to E-SEAS in battery-dependent only and non-battery-dependent only environments. In WR's case, the obtained results show that this algorithm has a more consistent throughput gain in hybrid environments (between 20% and 40%). However, with homogeneous device-type environments (battery-dependent only and non-battery-dependent only), the behavior was better only when the job load was 1500 jobs assigned and E-SEAS obtained better results in other situations. The overall results show that we can have better throughput with energy-aware scheduling algorithms in the hybrid environment, especially when the job load is not heavy.

Figure 5 indicates that, in a hybrid environment, E-SEAS job assignments are all to non-battery-dependent devices.

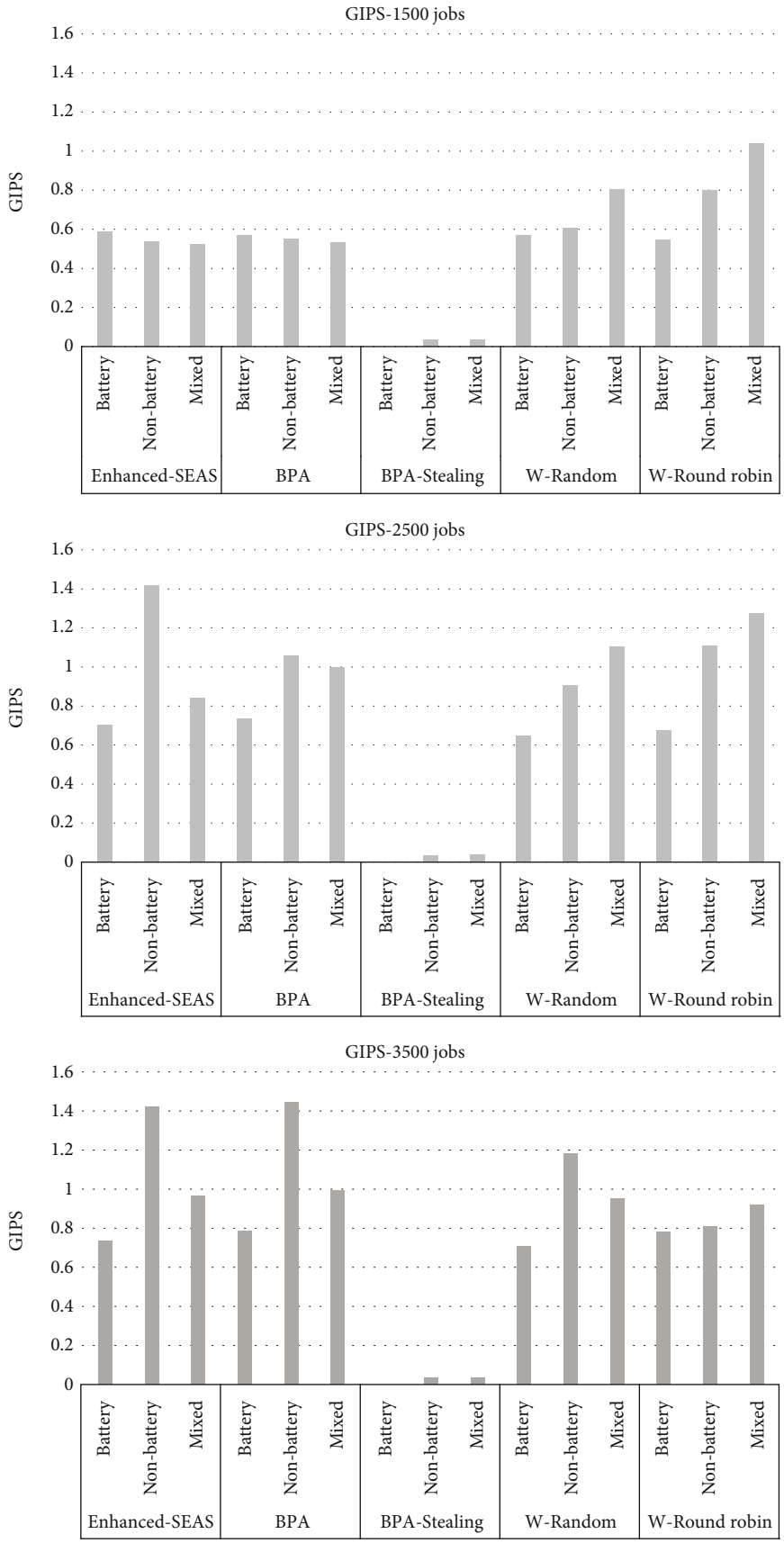

Figure 4: System giga operations per second.

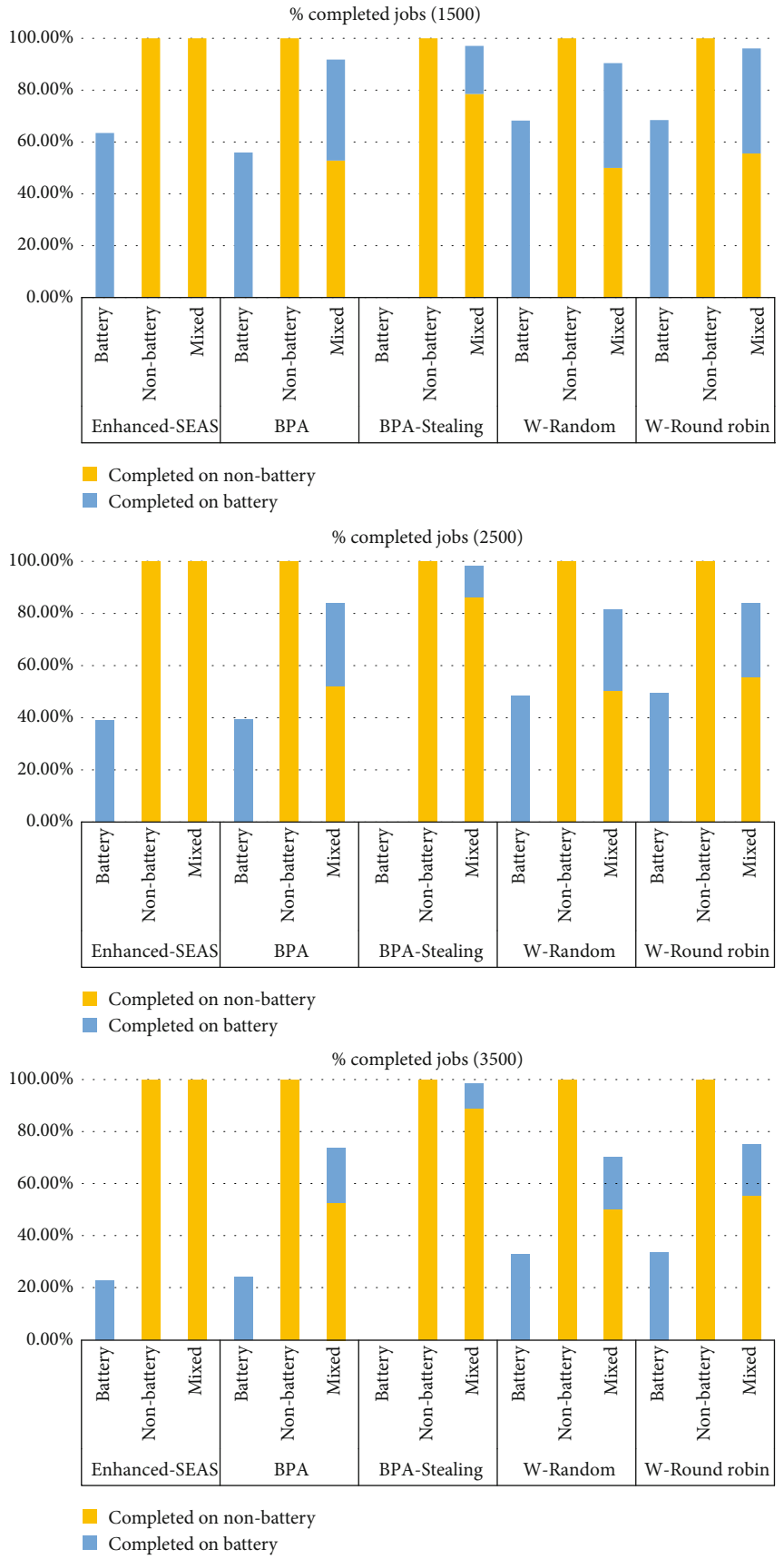

Figure 5: System job completion rate.

The node ranking logic cannot distinguish between the non-battery- and battery-dependent devices. This behavior can be explained using the main logic of E-SEAS. E-SEAS assigns jobs depending on the estimated uptime and the device's capabilities. However, as in the case of hybrid environments, the non-battery devices report that their uptime will be infinite; they always get a better rank compared to the battery devices. That is, the battery devices never got a chance to get selected.

On the other hand, for a hybrid environment, we can see that BPA obtained between 70% and 92% of the job completion rate, and WR obtained between 75% and 90% of the job completion rate. Finally, WRR obtained between 75% and 95% of the job completion rate. In battery-dependent environments, we can see that BPA, WR, and WRR obtained between 25% and 68% of the job completion rate, which is slightly better than the E-SEAS job completion rate. Putting both results together, we can see that BPA, WR, and WRR have a better completion rate, resulting in more processed jobs and, in some cases, better network performance. That is, these algorithms are scheduling and finishing more jobs without sacrificing network performance.

In the case of BPA with Job-Stealing, we got a peculiar result. This algorithm has the best completion rate of all the other algorithms, so this technique has the best awareness of the different kinds of devices present in the system. However, this technique shows the worst throughput in the experiments. We observed that this technique raises the network requirements due to the constant communication between nodes and permanent job transferring. This high traffic causes a bottleneck in the network affecting the overall performance of the system.

6. Conclusions

In this paper, we explored the capability of state-of-the-art schedulers. We proposed new ones to exploit the cooperation among low-powered battery- and non-battery-dependent devices to accomplish resource-intensive jobs at the Edge, i.e., exploitation of hybrid environments. The obtained results in the simulation reveal that our proposed algorithms, BPA, WR, and WRR, can obtain up to a 90% increment in overall throughput and around 95% of completed jobs in hybrid environments. We also showed that incorporating these characteristics gives more awareness of the type of resources present and can enable the algorithms to manage resources more efficiently in hybrid environments than other algorithms found in the literature. We also showed that Job-Stealing could contribute to better job distribution by taking advantage of idle devices in the network. However, it can only be used in scenarios where performance is not essential. In this way, in the future, we must address the performance problem by improving the network traffic and optimizing the communication between nodes.

We expect to improve and polish these proposed scheduling algorithms and incorporate them into real scenarios outside of a simulator in future work. We also expect to test these algorithms using more powerful non-battery-dependent devices to test their behavior in scenarios where these kinds of devices are more common in topologies based on Edge computing.

Data Availability

Available code and simulated environments are available in https://github.com/psanabriaUC/mobile-grid-simulator.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was funded by the National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL/2020-21200979, the National Agency for the Promotion of Research, Technological Development and Innovation (ANPCyT)/PICT-2018-03323, and the National Scientific and Technical Research Council (CONICET)/PIP 2017-2019 GI 11220170100490CO.

References

[1] Cisco, "Cisco Visual Networking Index: global mobile data traffic forecast update, 2017–2022 white paper - Cisco," 2019. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html (accessed Mar. 04, 2019).

[2] J. Feng and K. Yu, "Moore's law and price trends of digital products: the case of smartphones," *Economics of Innovation and New Technology*, vol. 29, no. 4, pp. 349–368, 2020.

[3] M. Hirsch, C. Mateos, J. M. Rodriguez, and A. Zunino, "DewSim: a trace-driven toolkit for simulating mobile device clusters in dew computing environments," *Software: Practice and Experience*, vol. 50, no. 5, pp. 688–718, 2020.

[4] M. Longo, M. Hirsch, C. Mateos, and A. Zunino, "Towards integrating mobile devices into dew computing: a model for hour-wise prediction of energy availability," *Information*, vol. 10, no. 3, p. 86, 2019.

[5] J. I. Benedetto, L. A. González, P. Sanabria, A. Neyem, and J. Navón, "Towards a practical framework for code offloading in the Internet of Things," *Future Generation Computer Systems*, vol. 92, pp. 424–437, 2019.

[6] M. S. Qureshi, M. B. Qureshi, M. Fayaz, M. Zakarya, S. Aslam, and A. Shah, "Time and cost efficient cloud resource allocation for real-time data-intensive smart systems," *Energies*, vol. 13, no. 21, p. 5706, 2020.

[7] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 393–413, 2014.

[8] Y. Wang, I. R. Chen, and D. C. Wang, "A survey of mobile cloud computing applications: perspectives and challenges," *Wireless Personal Communications*, vol. 80, no. 4, pp. 1607–1623, 2015.

[9] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: can offloading computation save energy?," *Comput*, vol. 43, no. 4, pp. 51–56, 2010.

[10] A. Tsipis, K. Oikonomou, V. Komianos, and I. Stavrakakis, "Performance evaluation in cloud-edge hybrid gaming

systems," in *Third International Balkan Conference on Communications and Networking 2019*2019 https://www.researchgate.net/publication/334883328.

[11] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[12] W. Yu, F. Liang, X. He et al., "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[13] R. Olaniyan, O. Fadahunsi, M. Maheswaran, and M. F. Zhani, "Opportunistic edge computing: concepts, opportunities and research challenges," *Future Generation Computer Systems*, vol. 89, pp. 633–645, 2018.

[14] S. Aslam, M. P. Michaelides, and H. Herodotou, "Internet of Ships: a survey on architectures, emerging applications, and challenges," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9714–9727, 2020.

[15] M. Hirsch, C. Mateos, and A. Zunino, "Augmenting computing capabilities at the edge by jointly exploiting mobile devices: a survey," *Future Generation Computer Systems*, vol. 88, pp. 644–662, 2018.

[16] M. Hirsch, C. Mateos, and A. Zunino, "Practical criteria for scheduling CPU-bound jobs in mobile devices at the edge," in *Proceedings -2018 IEEE International Conference on Cloud Engineering, IC2E*, pp. 340–345, Orlando, FL, USA, 2018.

[17] M. Hirsch, J. M. Rodriguez, A. Zunino, and C. Mateos, "Battery-aware centralized schedulers for CPU-bound jobs in mobile grids," *Pervasive and Mobile Computing*, vol. 29, pp. 73–94, 2016.

[18] M. Hirsch, C. Mateos, J. M. Rodriguez, A. Zunino, Y. Garí, and D. A. Monge, "A performance comparison of data-aware heuristics for scheduling jobs in mobile grids," in *2017 43rd Lat. Am. Comput. Conf. CLEI*, vol. 2017, pp. 1–8, Orlando, FL, USA, 2017.

[19] S. Bera, S. Misra, and J. J. P. C. Rodrigues, "Cloud computing applications for smart grid: a survey," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1477–1494, 2015.

[20] K. Habak, M. Ammar, E. W. Zegura, and K. A. Harras, "Workload management for dynamic mobile device clusters in edge femtoclouds," in *2017 2nd ACM/IEEE Symposium on Edge Computing, SEC*, vol. 14, pp. 1–14, San Jose/Fremont, CA, USA, 2017.

[21] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "An online algorithm for task offloading in heterogeneous mobile clouds," *ACM Transactions on Internet Technology*, vol. 18, no. 2, pp. 1–25, 2018.

[22] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 64–71, 2017.

[23] A. Mtibaa, A. Fahim, K. A. Harras, and M. H. Ammar, "Towards resource sharing in mobile device clouds," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 51–56, 2013.

[24] B. Li, Y. Pei, H. Wu, and B. Shen, "Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds," *The Journal of Supercomputing*, vol. 71, no. 8, pp. 3009–3036, 2015.

[25] L. Chunlin and L. Layuan, "Exploiting composition of mobile devices for maximizing user QoS under energy constraints in mobile grid," *Information Sciences*, vol. 279, pp. 654–670, 2014.

[26] M. N. Birje, S. S. Manvi, and S. K. Das, "Reliable resources brokering scheme in wireless grids based on non-cooperative bargaining game," *Journal of Network and Computer Applications*, vol. 39, no. 1, pp. 266–279, 2014.

[27] S. W. Loke, K. Napier, A. Alali, N. Fernando, and W. Rahayu, "Mobile computations with surrounding devices," *ACM Transactions on Embedded Computing Systems*, vol. 14, no. 2, pp. 1–25, 2015.

[28] S. C. Shah, "Energy efficient and robust allocation of interdependent tasks on mobile ad hoc computational grid," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 5, pp. 1226–1254, 2015.

[29] R. Buyya and M. Murshed, "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13–15, pp. 1175–1220, 2002.

[30] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[31] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: an environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, 2018.

[32] H. Flores, P. Nurmi, S. Tarkoma, and P. Hui, "Poster: MobileCloudSim: a context-aware simulation toolkit for mobile computational offloading," in *UbiComp/ISWC 2018 - Adjunct Proceedings of the 2018 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2018 ACM International Symposium on Wearable Computers*, pp. 38–41, Singapore, 2018.

[33] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: a toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[34] D. N. Jha, K. Alwasel, A. Alshoshan et al., "IoTSim-Edge: a simulation framework for modeling the behavior of Internet of Things and edge computing environments," *Software: Practice and Experience*, vol. 50, no. 6, pp. 844–867, 2020.

[35] J. M. Rodriguez, A. Zunino, and M. Campo, "Mobile Grid SEAS: simple energy-aware scheduler," in *3rd High-Performance Computing Symposium. 39th JAIIO*, vol. 54, no. 2293, pp. 3341–3354, Buenos Aires, Argentina, 2010.