A performance comparison of data-aware heuristics for scheduling jobs in mobile Grids

Matías Hirsch and Cristian Mateos and Juan M. Rodriguez and Alejandro Zunino
ISISTAN-UNCPBA-CONICET. Tandil, Buenos Aires, Argentina
E-mail: `matias.hirsch@isistan.unicen.edu.ar`
Yisel Garí and David A. Monge
ITIC-UNCuyo & CONICET. Mendoza, Argentina

***Abstract*—Given mobile devices ubiquity and capabilities, some researchers now consider them as resource providers of distributed environments called mobile Grids for running resource intensive software. Therefore, job scheduling has to deal with device singularities, such as energy constraints, mobility and unstable connectivity. Many existing schedulers consider at least one of these aspects, but their applicability strongly depends on information that is unavailable or difficult to estimate accurately, like job execution time. Other efforts do not assume knowing job CPU requirements but ignore energy consumption due to data transfer operations, which is not realistic for data-intensive applications. This work, on the contrary, considers the last as non negligible and known by the scheduler. Under these assumptions, we conduct a performance study of several traditional scheduling heuristics adapted to this environment, which are applied with the known information of jobs but evaluated along with job information unknown to the scheduler. Experiments are performed via a simulation software that employs hardware profiles derived from real mobile devices. Our goal is to contribute to better understand both the capabilities and limitations of this kind of schedulers in the incipient area of mobile Grids.**

***Keywords*— Mobile Grid, Mobile devices, resource intensive applications, job scheduling**

I. Introduction

Worldwide popularity and increasing capabilities of mobile devices have led researchers to propose mobile device inclusion as first-class resource providers in distributed computing environments. Offloading for mobile devices [12], [22], which originally promoted moving heavy computations from mobile devices to fixed high-end infrastructures, now considers local arrangements of nearby mobile devices to execute such computations. This approach reduces network latency, reduces the energy cost of remote data communication, and avoids the monetary charges inherent to Cloud infrastructures usage [14], [25], [16]. Other authors also propose to seamlessly integrate mobile devices into existing Grid environments to increase available resources [8], [13], [19].

Mobile devices have particular features that should be considered for integrating them in distributed computing environments [19]. These features include finite energy supply, ability to change location, and (wireless) unstable communication. Hence, to improve mobile device resource exploitation, prior works [8], [21], [18], [9] have shown the benefits of schedulers that consider such particular features. Another factor that these schedulers consider is the topology representing the underlying stack of communication technologies used to group/coordinate resources. Ad-hoc and proxy-based networks are the most targeted topologies. In the former, nodes reachability depends exclusively on the mobile nodes that integrate the network: nodes typically play a dual role, i.e., as end hosts and routers, forwarding packets wirelessly towards other mobile nodes that might not be within the direct transmission range of the sender. In proxy-based networks, nodes maintain single-hop wireless connections to fixed node called *proxy*. Offering local resources behind a proxy to a Grid infrastructure is a strategy commonly adopted by traditional Grid platforms like Ibis-Satin [24], JCluster [27] and GridGain[1]. Due to its simplicity, proxy-based networks have been the starting point for defining different local scheduling policies in mobile Grids [7], [2], [5], [18]. Figure 1 depicts a proxy-based mobile Grid which is the targeted topology in this work. When jobs are submitted to the system, a scheduler, which might operate either in online or batch mode, assigns these jobs to reachable nodes.

The distinct special mobile devices feature subsets, together with the universe of job and resources available information assumed and objectives of the associated resource allocation problem [7], result in a large number of challenging scheduling scenarios for which new schedulers must be investigated. As far as we know, most of these scenarios have not been explored in the literature yet. This work targets one of these broad scenarios by conducting a study of several traditional heuristics for scheduling independent jobs whose CPU-time requirements are unknown and data transferring requirements are known by the scheduler. Then, this study presents an overview of the effectiveness in energy utilization when scheduling is performed based on user-provided data-transfer job information. Energy utilization in a mobile Grid using the different heuristics is measured via system throughput, using an existing Java-based simulation software of our own [7], [8].

The organization of this paper is as follows. In the following Section we discuss related efforts and list our contributions compared to such efforts. Section III describes the energy-aware, data-oriented scheduling heuristics studied and evaluated in this work. Particularly, Section IV describes the experiments done and results obtained. Lastly, Section V concludes the work and briefly delineates future research.

978-1-5386-3057-0/17/$31.00 ©2017 IEEE

[1]http://www.gridgain.com

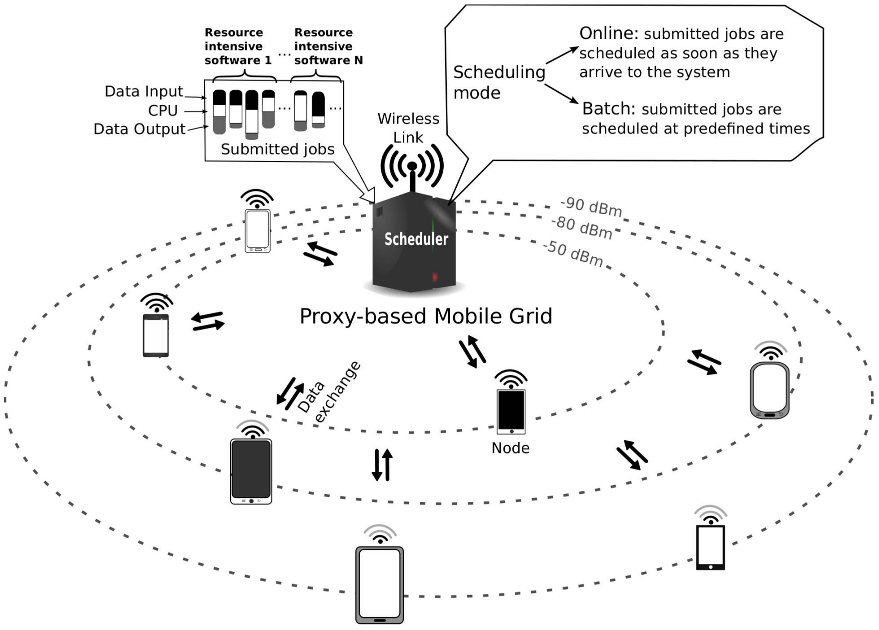
Resource intensive software 1
Resource intensive software N
Data Input
CPU
Data Output
Submitted jobs
Scheduling mode
Online: submitted jobs are scheduled as soon as they arrive to the system
Batch: submitted jobs are scheduled at predefined times
Wireless Link
Scheduler
-90 dBm
-80 dBm
-50 dBm
Proxy-based Mobile Grid
Data exchange
Node

Fig. 1. Overview of a proxy-based mobile Grid

II. Background

Wireless infrastructures to which people connect with mobile devices are present in an increasing number of daily life public and private places. Coffee shops, restaurants, shoppings, university campuses, work offices, just to mention few examples, are equipped with wireless access points to let the people surf the Web, check emails, read a newspaper, play online games, or simply stay connected for receiving messages and notifications of their interest. Such contexts fit the proxy-based topology mentioned, and therefore via proper scheduling mechanisms, the joint computing capabilities of mobile devices could be used to solve complex computational problems.

Despite the incipient nature of the area, several works address job scheduling using mobile devices as resource providers [14], [5], [2], [16], [25]. They assume complete knowledge, or accurate estimations, about all jobs requirements from the software to run and resources characteristics. Then, it is common that these works propose different optimal or near-optimal schedulers with constraints that are associated to the mobile device special features and/or topologies introduced above. For instance, [5], [2], [16], [25] treat the limited energy of mobile devices as a formal resource constraint. Moreover, [14] and [16] focus on optimizing energy utilization in ad-hoc multi-hop networks.

Other schedulers [15], [8], [21], [18] do not assume hard knowledge of job requirements as the previous set, but are designed to deal with only one kind of job, i.e., CPU-intensive [15], [8], [18] or data-intensive [21]. For instance, [7], [8] target CPU-intensive jobs and take into account the limited energy of mobile devices as they consider the last state of charge reported by devices and job energy consumption indicators derived from device benchmarks to perform the job assignments. In [21], a high-level job classification is used to select the most appropriate group of nodes taking into account their probability of staying connected and the energy-related properties of their communication paths. While [15], [21] target mobile ad-hoc networks, [7], [8] target proxy-based networks.

This work differs from those mentioned above in that:

- We target a *hybrid* type of job requirements, meaning that neither job data transfers nor job CPU times are negligible in terms of energy consumption. Although job CPU requirements are not negligible, the heuristics studied operate mainly based on the sizes of job input and output data to be transferred before and after the job execution, respectively. From a user's perspective, this information is easier to specify in real-life settings than CPU time because for the latter, numerous variables should be taken into account, e.g., the compiler used to build the job binary, the hardware where the job will run, the system load at the time the job is executed, etc. Besides, to predict the time an arbitrary code will take to run requires knowing whether it will ever finish its execution, i.e., solving the Halting problem [26].
- We introduce three data-energy aware heuristics, inspired on traditional scheduling techniques, plus three genetic algorithms, and evaluate them varying job data and CPU time requirements. Then, we aim to study the throughput achieved by the heuristics when only data-related job information is available to the scheduler. We have chosen genetic algorithms due to their versatility and performance to solve combinatorial optimization problems in many domains, including job scheduling.
- We compare the performance of these heuristics with that of E-SEAS [8], an online scheduler for CPU-bound jobs that shares the principles of the SEAS [18] and has been regarded as an efficient scheduler by third-party

researchers [5]. Details on E-SEAS are provided in the next Section.
- Rather than using synthetic mobile device energy consumption data, we simulate energy consumption of mobile Grid nodes via CPU usage profiles extracted from real mobile devices, and data transferring information arisen from exhaustive past studies [6], [17], [1] that characterize energy consumption in mobile devices due to network usage.

III. Data-aware scheduling heuristics in proxy-based mobile Grids

Traditional scheduling heuristics, e.g., Min-min, Max-Min and MCT (Minimum Completion Time), have been extensively studied in fixed computational Grids and also used as baseline for comparison [23], [11]. The performance of these heuristics is usually measured via time-oriented metrics like makespan and flowtime. To apply these traditional heuristics, it is necessary to know the completion time of every job on every candidate computing node. For CPU-intensive jobs, such time arises from the amount of CPU cycles that a node should perform to finish the execution of a job, which at least depends on the hardware characteristic and current load of the node that executes the job. When nodes of the distributed computing environment present heterogeneous hardware and load, each job needs to be associated a running time for each candidate node. This information is commonly represented by an ETC matrix where rows represent jobs and columns represent nodes [10].

The completion time of an hybrid job requirement –those heavily using both CPU and network resources, i.e., like the ones assumed in this work– is composed by an execution time and a data transferring time –for job input and output data–. However, it is not always possible to feed the scheduler with information of jobs execution time because this requires, in the best case, job historical runs information that is not always consistent. Then, we assume the scenario where only job data transfer time is known by the scheduler. In fact, since we aim to measure efficiency in energy utilization –which is critical considering that mobile devices are battery-powered nodes–, we assume that the scheduler knows the energy cost of transferring jobs data. This goal on the other hand hinders the application of traditional heuristics in mobile Grids since, as explained, their performance is measured via time-oriented metrics. As a consequence, we adapt the way these traditional heuristics operate to represent resource utilization in energy-based instead of time-based units.

The adapted scheduling heuristics result in schedulers that are provided with jobs data transfer energy consumption cost derived from user provided job data-related information expressed in bytes. Such conversion is possible thanks to the characterization performed by several studies of mobile devices energy consumption in transferring data [6], [17], [1]. Particularly, the studies of [6], [20], show a direct relation between the Received Signal Strength Indicator (RSSI) and the energy consumed while transferring data that reveals that transfers with poor signal strength requires more energy than transfers with good signal strength. Moreover, the RSSI value is information that can be known by the scheduler through modern mobile OS APIs[2][3], which facilitates the practical implementation of the scheduling logic. Then, energy consumption of every job on every node of the mobile Grid can be estimated. To our knowledge, there is no published work utilizing such information to optimize energy utilization when scheduling hybrid jobs in a mobile Grid.

With the proposed resource utilization unit change, the ETC (expected time to compute) matrix employed by traditional heuristics is now an EET (expected energy to transfer) matrix, from which we derive the following adapted heuristics for handling jobs arriving to the proxy:

- The **Min-MinMobiComEnergy** batch heuristic specializes Min-Min. From the list of unassigned jobs, it selects those with the smallest aggregated input and output data to be transferred. Then, it selects the node whose remaining energy is the least affected by the transferring of the selected job plus the transferring of all previous job assignments to that node. Later, it adds the selected job to the list of assigned jobs of the corresponding node, removes it from the list of unassigned jobs and repeats the steps for the remaining jobs in that list until all jobs are processed or there is no node with enough remaining energy to transfer the next selected job.
- The **Max-MinMobiComEnergy** batch heuristic is an adaptation of the traditional Max-Min heuristic that operates similarly to Min-MinMobiComEnergy but the biggest aggregated input and output data size is used as selection criterion of the next job to assign.
- The **Remaining Transfer Capacity (RTC)** heuristic, is inspired in the online MCT heuristic. RTC immediately assigns the next incoming job to the node whose remaining transfer capacity is the least affected. At the time the remaining transfer capacity of a node is estimated, all future job output data transfers from previous job assignments, are considered.

In this work we also assess the **Enhanced Simple Energy-aware Scheduler (E-SEAS)** heuristic [7], which is not inspired in a traditional heuristic but it is an online scheduler specially designed for scheduling CPU-bound jobs in mobile Grids. To decide the most appropriate mobile node for executing a CPU-intensive job, the E-SEAS ranks all candidate nodes using a formula that combines their computing capability, current assigned jobs and battery State Of Charge (SOC). Every new incoming job is assigned to the node with the highest rank. The rank is recalculated upon every new job arrival. The equation 1 shows the formula that E-SEAS uses to rank nodes.

$$nodeRank = \frac{SOC * flops}{assignedJobs + 1} \tag{1}$$

In the equation, the SOC component is an integer value in the range 1 and 100 that represents the last battery SOC

[2]https://developer.android.com/reference/android/net/wifi/WifiInfo.html#getRssi()

[3]https://developer.apple.com/reference/corewlan/cwinterface/1426414-rssivalue

update sent from the mobile node to the proxy node. The $flops$ component is a pre-computed and indexed positive float value that represents the computing capability of the mobile node. It is obtained by means of running a mobile application[4] implementing the Linpack benchmark, which is designed to measure the float-point operations per second used to solve a system of linear equations. The *assignedJobs* component represents the number of jobs being executed and queued by the mobile node. To avoid getting a division by zero error, a value of one is added to *assignedJobs*. It is worth noting that Linpack is used to benchmark many supercomputers around the world included in the well-known Top500 list.

Batch traditional heuristics have served as baseline for other schedulers in traditional Grid environments [23], [11]. Therefore, we include in our performance study three Genetic Algorithms (GA). GAs have been successfully applied to job scheduling problems in fixed computational Grids [23], [11]. The GAs derived, combine parameter values and operators from a preliminary study where we explored ninety six combinations of different termination conditions, selection operators and variation operators -including different recombination probabilities p_c and mutation probabilities p_m-. Table I shows

TABLE I
GA PARAMETERS SET SELECTED

GA parameters	Value
Termination condition	30 seconds for fitness improvement, or maximum of 5 minutes of evolution
Parent selection	Tournaments of 10 individuals with replacement
Recombination	UniformCrossoverOperator with p_c of 0.8
Mutation	RandomMutationOperator with p_m of 0.15
Population replacement	Deterministic crowding

the parameters combination which achieved the highest fitness value within the shortest time. The maximum execution time of all proposed GA versions was set to five minutes. Such a short time window was chosen because during the GAs execution time, mobile devices, which are connected to a proxy and ready to receive jobs, consume energy from their batteries related to the maintenance of an established WiFi connection and the base consumption of the mobile OS.

The three GAs basically differ in their initial populations of individuals. An individual is a solution to the scheduling problem. The GA versions named GA_Min-MinMobiComEnergy and GA_Max-MinMobiComEnergy include Min-MinMobiComEnergy and Max-MinMobiComEnergy individuals within their initial populations respectively. Besides, the rest of individuals that complete the population size used (100) are randomly generated by not exceeding the amount of assigned jobs contained in the individual that represents the Min-MinMobiComEnergy or Max-MinMobiComEnergy heuristic depending on the GA version. The GA version named GA_random was created with all random individuals.

[4]https://play.google.com/store/apps/details?id=com.sqi.linpackbenchmark

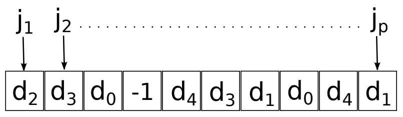

Fig. 2. Adopted encoding: Example

We represent individuals as array of integers. Figure 2 shows an example. The genes of an individual (array positions) are job identifiers, while alleles (array values) are mobile device identifiers. The value -1 indicates that the job is not assigned to a device yet. Moreover, the scheduling problem that is addressed by the GAs is formally defined as follows. Given:

- $D = \{d_1, d_2, ..., d_n\}$, is the set of devices of the mobile Grid,
- $S = \{s^{d_1}, s^{d_2}, ..., s^{d_n}\}$, is the set of signal strength values from $d_1, d_2, ..., d_n$, respectively, at the time jobs scheduling starts,
- $E = \{e^{d_1}, e^{d_2}, ..., e^{d_n}\}$, is the available Joules of devices at the time jobs scheduling starts,
- $J = \{j_1, j_2, ..., j_p\}$, is the set of job data requirements to be scheduled, where $j_{q_{id}}$ are the #bytes of input data of job j_q, and $j_{q_{od}}$ are the #bytes of output data of job j_q,
- $f_{(b,s)}$ is a function that returns the Joules consumed for transferring b bytes with a received signal strength value s.

The goal is to find device assignments $A = \{a_1, a_2, ..., a_n\}$ with each device assignment a_u represented by the pair $< d_u, J^{d_u} >$ where $d_u \in D$, $J^{d_u} = \{j_r^{d_u}, j_s^{d_u}, j_t^{d_u}, ...\} / J^{d_u} \subseteq J$, $J^{d_1} \cap J^{d_2} \cap ... \cap J^{d_n} = \emptyset$ and $J^{d_1} \cup J^{d_2} \cup ... \cup J^{d_n} \subseteq J$, such that $Max(fitness)$ where the $fitness$ of device assignments A used by the GAs for measuring a solution quality is defined as:

$$fitness = (1 - CPUBalance(A)) + \frac{TotalTransferedJobs(A)}{p} \quad (2)$$

being

$$CPUBalance(A) = \frac{\sqrt{\sum_{i=1}^{n} \left(d_{i,currAssign} - d_{i,expectedAssign}\right)^2}}{\sqrt{\sum_{i=1}^{n} \left(d_{i,WorstAssign} - d_{i,expectedAssign}\right)^2}} \quad (3)$$

a value within $[0,1]$ which assesses the CPU load balancing of device assignments A. This value is the Euclidean n-space distance between the vector that represents device assignments A (*currAssign*) and the vector of device assignments resulting from the application of a CPU-bound job criterion (*expectAssign*). Due to its simplicity and effectiveness, the E-SEAS [8] is employed as criterion to rate processing capability of nodes and balance the energy consumption due to jobs CPU usage. The E-SEAS selects the most appropriate device to execute a CPU-bound job based on device FLOPS, remaining battery charge and currently assigned jobs. To obtain *expectedAssign* vector, E-SEAS is iteratively applied to the list of jobs assigned in A. To provide a normalized $CPUBalance(A)$, the resulting distance between *currAssign* and *expectedAssign* vectors is divided by the maximum distance value which is computed as the euclidean n-space distance between the vector that represent the worst possible job-device combinations (*worstAssign*) and *expectedAssign*. The worst device assignments arises from assigning all jobs to the device that, according to the CPU-bound job criterion, should receive the least amount of jobs.

Moreover, $TotalTransferedJobs(A)$, the amount of fully transferred jobs, is the sum of completed transferred jobs of each device assignment, which, in turn, is defined by $TransferedJobs(a_u) = \sum_{k=1}^{\#j \in J^{d_u}} fit\left(k, J^{d_u}\right)$, where $fit\left(k, J^{d_u}\right)$, defined by Eq. 4, determines whether d_u has enough energy to fully transfer the k-th job of the device assignment a_u.

$$\begin{cases} 0 & if \sum_{x=1}^{k} enTranf_{j_x^{d_u}} > e^{d_u} \\ 1 & if \sum_{x=1}^{k} enTranf_{j_x^{d_u}} \leq e^{d_u} \end{cases} \tag{4}$$

where $enTranf_{j_k^{d_u}} = f\left(j_{k_{id}}^{d_u}, s^{d_u}\right) + f\left(j_{k_{od}}^{d_u}, s^{d_u}\right)$ is the energy consumed by the device d_u for receiving $j_{k_{id}}$ job input from the proxy node and for sending $j_{k_{od}}$ job output to the proxy node. The formulation aims at maximizing the energy utilization by targeting the highest amount of jobs completed.

IV. Evaluation

A. Methodology

The experimental methodology was simulation, which is an accepted practice in distributed computing [3], [4]. It facilitates, i.e., provides a repeatable and controllable manner of evaluating scheduling techniques in distributed environments of high heterogeneous resources and dynamic resource availability, e.g., Grids, Mobile Grids and Clouds. We employ a Java-based event-driven *simulator* [7], [8] that models everything that might occur in a mobile Grid (e.g., jobs arrivals, jobs completion, devices battery drop, network activity derived from jobs data input/output transferring and devices status notifications) via events.

The events that simulate how devices residual energy decreases in time are generated by combining energy consumption information from CPU usage profiles and WiFi usage measurements. Both, profiles and measurements, were extracted from real mobile devices (smartphones and tablets). A CPU usage profile reflects the rate at which a device energy is consumed under certain CPU usage. The CPU usage profiling is a pre-simulation procedure performed with an application installed in real mobile devices whose implementation details can be found in [7]. In short, a profile starts with a full charged device and ends when the device shuts down due to battery depletion. During that time, the profiling application logs time-stamp, battery SoC, and CPU usage information. By means of another application, a profile is converted into two lists of chronologically ordered events that serve as input for the simulator. One list contains battery drop events and the other CPU events. The former is used to simulate the energy consumption and available energy of a device while the latter is used to simulate the execution time of job based on the node available CPU. Both the simulator and the profiling application (for Android) are available at https://github.com/cmateos/mobileGridSimulator.

To contemplate time and energy consumption derived from networking activity, we re-utilized findings of exhaustive and focused third-party studies [6], [17], [20], particularly those related to mobile-to-fixed-node WiFi communication because it is the form of communication adopted in the proxy-based topology. From information and wireless networking theory, it is known that energy consumption and time of wireless data transferring is influenced by numerous variables such as distance attenuation, shadowing by obstacles, channel interference, transport protocol, among others. However, most of these variables are not accessible nor controlled from the application layer where the job scheduling logic operates in real mobile Grids. Then, we contemplate such variations supported on properties of receive signal strength indicator (RSSI) [6], [20] that suggest that with poor RSSI the energy consumption and time of transferring data increases exponentially. Then, we simulate the energy consumption and time for transferring data as functions of RSSI value which can be obtained through devices OS API.

The technical details of how energy consumption information from CPU profiles and network measurements are aggregated to reflect devices residual energy during a simulation are described as follows. The details are given for one device but describe the behavior of all simulated devices. The battery drop events from CPU usage profile define the baseline energy consumption. This is because the CPU energy consumption is always present while the device is operative. From the scheduling point of view, the CPU of a device is considered to be in two possible states, i.e., idle or in service, and there are distinct CPU profiles to represent each state. An idle CPU usage profile reflects the energy consumed by a device that executes processes or tasks related to the operating system, while an in service CPU usage profile reflects the energy consumed by a device that is executing a job sent by the proxy of the mobile Grid.

In every simulation step, a device has an “in current use” CPU usage profile that changes on every CPU state change. Notice that a CPU profile has a battery drop event list associated that changes with every CPU state change. For instance, when a device starts the execution of a job, its “in current use” CPU usage profile switches from idle to in service. By contrast, when a device finishes a job execution, its “in current use” CPU usage profile switches from in service to idle.

Irrespective of the in current use CPU usage profile, the battery drop events of the profile only reflect the device energy consumption caused by CPU usage. To incorporate energy consumption derived from networking activity, the information of battery drop events is adjusted with every event that involves a data transferring, e.g., when jobs input/output is transferred, battery updates are informed from the mobile node to the proxy, etc. Figure 3 shows a graphical representation of different adjustments applied to baseline battery drop events. In both cases, the dash line shows how device residual energy would decrease in absence of the networking event. In contrast, the filled line that follows the networking event shows how baseline battery drop events are brought forward in time as consequence of the adjustment. In other words, the effect of

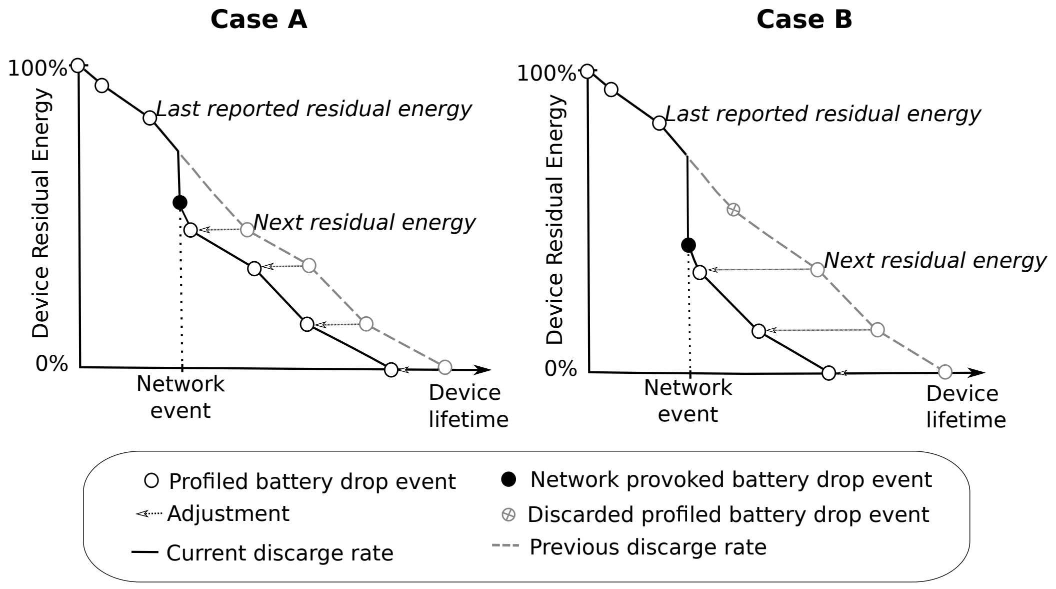

Fig. 3. Adjustments during simulation of battery drop events as consequence of networking activity

any adjustment is reflected with a shortening of the device lifetime. An adjustment involves the following steps:

1) Compute the energy spent by the networking event: the amount of data to be sent or received by the device and its RSSI value are used to compute the energy spent in the networking event.
2) Reflect the networking event on the current residual energy: the residual energy at the time the networking event occurs, i.e., the current residual energy, is calculated by evaluating the linear function that contains the last reported residual energy and the next residual energy. Then, such value is updated by subtracting the energy consumption resulted from step 1.
3) Define the adjustment: to perform the translation of future battery events in time, i.e., the adjustment, it is necessary to determine the equation of the linear function that joins the last reported residual energy with the residual energy resulted from step 2.
4) Reflect the networking event on the device lifetime: this means adjusting the time in which the next battery drop events will occur by using the equation of step 3.

The difference between case A and case B is that the adjustment of case A does not involve a networking energy consumption that exceeds the residual energy of the next battery drop event while case B does. In case B, the battery drop events whose residual energy is less than the new residual energy, i.e., the one calculated in step 2, are discarded as future events of the device simulated lifetime.

B. Experiments setup and results

The used topology had 100 mobile devices connected to a fixed proxy with varied RSSI values and hardware characteristics –i.e., FLOPS, battery capacity, CPU battery consumption profiles– obtained from real mobile devices. We simulated a mobile Grid with 20 tablets Acer A100, 30 tablets Samsumg Galaxy Tab 2, and 50 smartphones LG L9, with RSSI values from -90 dBm to -50 dBm. The smaller the dBm value, the higher the energy cost per transferred byte.

Job data sets are composed by synthetic jobs whose input and output data vary in [1 - 500] MB (uniform probability) and CPU operations relates in $n * log\,n$, n^2 or n^3 to the input data size in KB. The three relations occur with the same probability and coexist in the same job set, meaning that jobs with almost equal input size can require different amount of CPU time. The range selected for job data is common nowadays in off-line navigation applications, games updates, etc, while the above complexities are present in real-life algorithms such as sorting, matrix multiplication, 3D matrix processing, respectively. Three job sets were generated using a continuous uniform distribution to ensure a controlled heterogeneity [8]. The job sets named *non-saturated*, *saturated* and *super-saturated* are composed of 2500, 3500 and 4500 jobs respectively. Job set saturation level is determined by the relation between total amount of input and output data -*job set data*- and the topology data transfer capacity. The latter, is defined as the sum of maximum amount of data each node is able to transfer with its initial energy -fully charged battery- and its RSSI value -which is assumed not to vary while the device is connected to the proxy-. When the *job set data* is less than the maximum transfer capacity of the topology the job set configuration is *non-saturated*. When it is greater, the configuration is *saturated*. When only the total jobs input data is greater, the configuration is *super-saturated*.

Figure 4 shows the percentage of completed jobs. Online heuristics are competitive in all job configurations, including the E-SEAS, which does not take into account data-related job information. The GA_random performed very poor in all scenarios, which shows the importance of starting the exploration of the solution space from a region with good individuals as the other two GA versions do. Moreover, the advantage of considering a dual component fitness function,

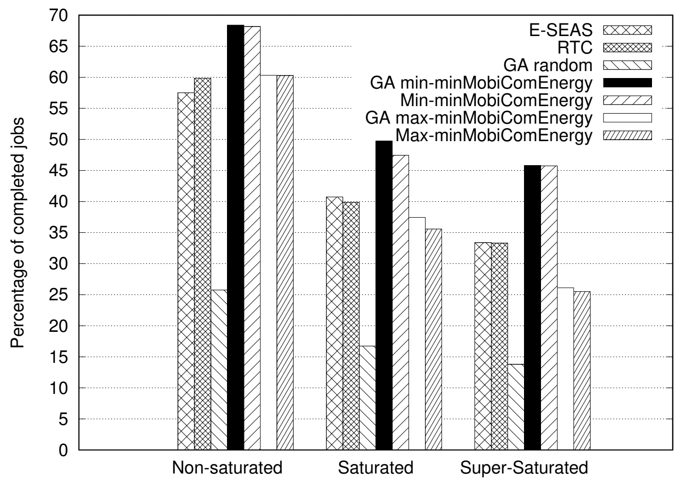

Fig. 4. Completed jobs (%) per job set

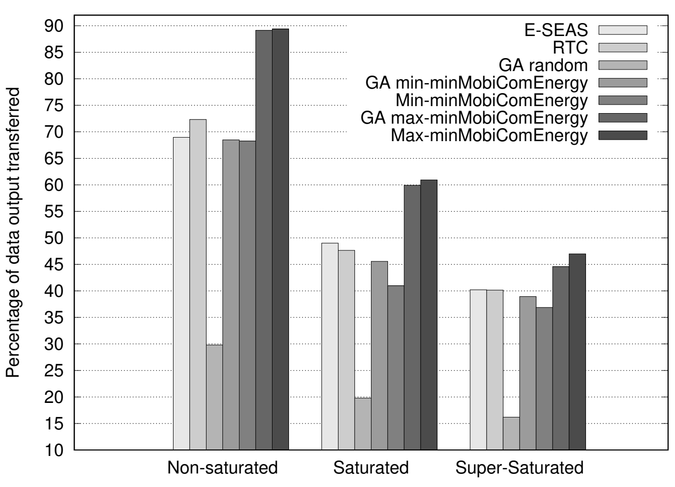

Fig. 5. Data output transferred (%) per job set

i.e., transfer and computing capabilities of devices, allowed GA_Min-MinMobiComEnergy and GA_MaxMinMobiComEnergy to complete a slightly higher amount of jobs than the corresponding heuristics used to seed the initial population. We complement these results with Figure 5, which shows the percentage of output data transferred (job results).

Output data transferred is important to measure since it is the last energy-consuming step a device performs to complete a job. From Figure 5 it can be seen that the heuristics which transfer the highest percentages of data output (GA_Max-minMobiComEnergy and Max-minMobiComEnergy) are within the group of those that achieved, according to Figure 4, the lowest percentages of completed jobs. The inverse behavior is manifested by Min-MinMobiComEnergy and its corresponding GA version. This suggests that Min-MinMobiComEnergy strategy completes more jobs than Max-MinMobiComEnergy because the first schedules data transfers of the smallest jobs before the largest jobs, which gives many smallest jobs the opportunity to be completed before the greatest ones.

When comparing the performance of GA versions (excluding GA Random) against that of the corresponding seed heuristic, the relationship between percentage of data output transferred and percentage of completed jobs does not hold. Precisely, this means that GA versions not only achieve slightly better performance in completing jobs than their corresponding seed heuristics, but also transfer more output data. Moreover, online heuristics performance approximates, in this case, data output percentages achieved by Min-MinMobiComEnergy and the corresponding GA version.

Furthermore, online heuristics –even E-SEAS, which does not take into account jobs data transfer information –achieve a competitive balance of completed jobs and data output transferred w.r.t. at least two batch heuristics. The potential to further improve online heuristics is however limited since jobs must be scheduled as soon as they are submitted to the scheduler, which prevents such heuristics from having a global picture of the total input/output data to be scheduled. Consequently, batch heuristics might be more advantageous since some time can be dedicated to optimize data transferring of a batch of jobs. Particularly, in regard to the studied GAs, the performance of GA_Min-MinMobiComEnergy and GA_Max-MinMobiComEnergy was slightly better than the performance of the heuristics used to seed the initial population, which should drive the focus of future improvements.

Note that GA_Min-MinMobiComEnergy and Min-Min-MobiComEnergy far exceed the jobs completed of all the other heuristics. This is due to scheduling small data transferring jobs first, leaves bigger "holes" of energy in mobile nodes which are eventually used by CPUs to produce jobs output.

V. Conclusions

We have studied seven heuristics for scheduling jobs on proxy-based mobile Grids, where only input and output data transfer job requirements are known to the scheduler. The experimental job sets included jobs of quite varying size in terms of CPU time ranging from few seconds to several hours in the most powerful device, and data requirements ranging from one to five hundred megabytes.

One finding is that when scheduling jobs with CPU and data requirements, both non-negligible in terms of energy consumption, the heuristics which take into account the provided job data-related information improves the performance over those who do not. Moreover, in practice, when jobs should be assigned as soon as they are submitted to the system, RTC is a viable option. However, if submitted jobs can wait some time until they are assigned to a node as in classical queued high-throughput environments, either GA_Min-MinMobiComEnergy and Min-MinMobiComEnergy are choices with quite balanced performance, in the sense that they achieve the highest percentage of completed jobs and competitive data output transferred. Achieve high throughput, in terms of completed jobs, is useful when the scheduler has to deal with jobs from multiple users. Moreover, when high throughput refers to prioritize job execution with high output data generation then GA_Max-MinMobiComEnergy and Max-MinMobiComEnergy are preferred.

A weak point of RTC, Min-MinMobiComEnergy and Max-MinMobiComEnergy is that, due to their algorithmic nature, they cannot be further improved. However, there are many variants/configurations for the different elements of the studied GAs that could be explored in the future. In this sense, we will extend our GAs with new mutation operators to allow jobs permutation and with this explore new areas of the solution landscape. Besides, we plan to further investigate the synergy between data and other CPU aware criteria [7]. We will also combine the exploration capability of GA with the exploitation capability of memetic algorithms. We also plan to assess how GAs respond in even more realistic scenarios where, e.g., where the RSSI values dynamically vary and/or mobile owners actively use their devices. Lastly, we will backup future experiments through statistical significance tests.

Acknowledgment

We acknowledge the financial support by ANPCyT through grants PICT-2012-0045 and PICT-2013-0464, and CONICET through grant PIP 11220120100185CO.

References

[1] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *9th ACM SIGCOMM conference on Internet measurement conference*, pages 280–293. ACM, 2009.

[2] 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*, 39:266 – 279, 2014.

[3] 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*, 14(13):1175–1220, 2002.

[4] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. 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*, 41(1):23–50, 2011.

[5] L. Chunlin and L. Layuan. Exploiting composition of mobile devices for maximizing user qos under energy constraints in mobile grid. *Information Sciences*, 279:654 – 670, 2014.

[6] N. Ding, D. Wagner, X. Chen, A. Pathak, Y. C. Hu, and A. Rice. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. *ACM SIGMETRICS Performance Evaluation Review*, 41(1):29–40, 2013.

[7] M. Hirsch, J. M. Rodríguez, C. Mateos, and A. Zunino. A two-phase energy-aware scheduling approach for cpu-intensive jobs in mobile grids. *Journal of Grid Computing*, 15(1):55–80, 2017.

[8] M. Hirsch, J. M. Rodríguez, A. Zunino, and C. Mateos. Battery-aware centralized schedulers for cpu-bound jobs in mobile grids. *Pervasive and Mobile Computing*, 29:73–94, 2016.

[9] K. Katsaros and G. C. Polyzos. Evaluation of scheduling policies in a mobile grid architecture. In *International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pages 390–397, 2008.

[10] J. Kołodziej. *Independent Batch Scheduling: ETC Matrix Model and Grid Simulator*, pages 19–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[11] J. Kołodziej, S. U. Khan, L. Wang, and A. Y. Zomaya. Energy efficient genetic-based schedulers in computational grids. *Concurrency and Computation: Practice and Experience*, 27(4):809–829, 2015.

[12] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava. A survey of computation offloading for mobile systems. *Mobile Networks and Applications*, 18(1):129–140, 2013.

[13] D. Lee, H. Lee, D. Park, and Y.-S. Jeong. Proxy based seamless connection management method in mobile cloud computing. *Cluster Computing*, 16(4):733–744, 2013.

[14] 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*, 71(8):3009–3036, 2015.

[15] S. W. Loke, K. Napier, A. Alali, N. Fernando, and W. Rahayu. Mobile computations with surrounding devices: Proximity sensing and multilayered work stealing. *ACM Transactions on Embedded Computing Systems*, 14(2):22:1–22:25, 2015.

[16] A. Mtibaa, A. Fahim, K. A. Harras, and M. H. Ammar. Towards resource sharing in mobile device clouds: Power balancing across mobile devices. *ACM SIGCOMM Computer Communication Review*, 43(4):51–56, 2013.

[17] A. Rice and S. Hay. Measuring Mobile Phone Energy Consumption for 802.11 Wireless Networking. *Pervasive and Mobile Computing*, 6(6):593–606, 2010.

[18] J. M. Rodríguez, A. Zunino, and M. Campo. Mobile grid seas: Simple energy-aware scheduler. In *3rd High-Performance Computing Symposium - 39th JAIIO*, 2010.

[19] J. M. Rodríguez, A. Zunino, and M. Campo. Introducing mobile devices into grid systems: a survey. *International Journal of Web and Grid Services*, 7(1):1–40, 2011.

[20] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. N. Padmanabhan. Bartendr: a practical approach to energy-aware cellular data scheduling. In *Sixteenth annual international conference on Mobile computing and networking*, pages 85–96. ACM, 2010.

[21] S. C. Shah. Energy efficient and robust allocation of interdependent tasks on mobile ad hoc computational grid. *Concurrency and Computation: Practice and Experience*, 2014.

[22] M. Sharifi, S. Kafaie, and O. Kashefi. A survey and taxonomy of cyber foraging of mobile devices. *IEEE Communications Surveys Tutorials*, 14(4):1232–1243, Fourth 2012.

[23] R. Subrata, A. Y. Zomaya, and B. Landfeldt. Artificial life techniques for load balancing in computational grids. *Journal of Computer and System Sciences*, 73(8):1176–1190, 2007.

[24] R. van Nieuwpoort, G. Wrzesinska, C. J. H. Jacobs, and H. E. Bal. Satin: A high-level and efficient grid programming model. *ACM Trans. Program. Lang. Syst.*, 32(3), 2010.

[25] X. Wei, J. Fan, Z. Lu, and K. Ding. Application scheduling in mobile cloud computing with load balancing. *Journal of Applied Mathematics*, 2013, 2013.

[26] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström. The worst-case execution-time problem - overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems*, 7(3):36:1–36:53, 2008.

[27] B.-Y. Zhang, Z. Mo, G. Yang, and W. Zheng. Dynamic load balancing efficiently in a large-scale cluster. *International Journal of High Performance Computing and Networking*, 6(2):100–105, 2009.