Scheduling Strategies for Mobile Devices in BOINC

Arturo García, Mariela Curiel

Facultad de Ingeniería. Pontificia Universidad Javeriana (PUJ) – Bogotá, DC – Colombia

{arturogarciap,mcuriel}@javeriana.edu.co

Abstract. *Executing tasks on mobile devices brings some challenges that must be addressed, such as device heterogeneity, limited CPU power, limited memory, short battery life, mobility, and intermittent disconnections. BOINC (Berkeley Open Infrastructure for Network Computing), the platform chosen in our research project that aims to process medical images using mobile phones, does not completely solve these challenges. In this sense, this article's objective is to present the modifications to BOINC's scheduling strategy to consider the characteristics of mobile devices and possible disconnections. The article describes BOINC-MGE (BOINC - Mobile Grid Extension), a BOINC extension that incorporates two new task scheduling algorithms to ensure the completion of tasks while efficiently using the mobile devices' energy. The strategies are tested using an image segmentation algorithm.*

Keywords: Mobile computing, BOINC, task scheduling, smartphones, volunteer computing, replication.

1. Introduction

The amount of people using mobile devices around the world has increased significantly in recent years. In particular, some authors consider that smartphones are one of the best inventions available for this generation. Its accelerated adoption suggests that this technology will continue to develop as users become aware of all the things they can do with only a couple of clicks. On the other hand, due to the recent advances in processors with low power consumption, devices such as tablets and smartphones can handle computationally intensive applications; hence, they could be considered the future computing platforms (Mengistu & Che, 2019). Some examples of the use of smartphones as computing platforms can be found in (Hirsch et al., 2017), (Duan et al., 2014), and (Arslan et al., 2015).

In the same address, several researchers have worked in incorporating mobile devices into the grid (e.g. (Furthmüller. & Waldhorst, 2010) (Hijab & Avula, 2011)). Particularly, a **Mobile Grid** is a specific grid where users can connect using their mobile devices (smartphones, tablets, etc.), mainly for two purposes: 1) to obtain access to the resources provided by the grid, and/or 2) to place their mobile devices as computing platforms available to all users of the traditional grid; being this one the use case which we are interested in our research.

The research project, from which derives the work presented in this article, is oriented to the distributed processing of medical images on mobile devices. In this project, BOINC was selected as the execution platform due to its complying with the following requirements: the possibility of running tasks on mobile devices using Android, support for the execution of programs written in C/C ++, the availability of its source code and technical documentation. C++ support was important since this is the language used by most of the imaging processing libraries.

BOINC is a platform for volunteer computing, and although it allows the use of mobile devices as execution platforms, some challenges of task execution are not fully resolved yet. These challenges are mainly due to user mobility, wireless connections, and limitations in battery power. These elements can cause intermittence, leading to an interruption of the running tasks and their subsequent rescheduling.

In this article, we describe the inclusion of two new scheduling algorithms in BOINC. These algorithms allow an adequate selection of mobile devices to complete the tasks since they consider the battery's current state and several users disconnect scenarios. Since we select the algorithms from those existing in the literature, we also describe the main types of scheduling strategies proposed in the area. The results obtained emphasize the importance of using scheduling strategies that consider mobile devices' characteristics when thinking about them as execution platforms.

The rest of this paper is structured as follows: Section 2 details the research's main concepts. Section 3 discusses some related work and how they differ or align with our research. Section 4 presents the new scheduling strategies for BOINC-MGE. Section 5 briefly describes how those strategies were included in the BOINC model. In Section 6, we present the experimental results obtained after comparing the native BOINC algorithms with the new ones. In Section 7, we discuss other options or features not covered in our research and how they can extend our work. Lastly, conclusions and future work are described.

2. Background

A basic explanation of the main concepts used in this article is provided in this section.

2.1 Grid Concepts

A **Grid** is a collection of heterogeneous distributed computing resources for solving large-scale computational and data-intensive problems. The rise of wireless technology and mobile devices has increased the number and types of resources to be integrated into the grid. As a result, the grid concept has been enriched, and new categories have emerged (Kurdi et al., 2008).

One of these new categories is the **Accessible Grid**, whose resources are available regardless of their physical capabilities and geographical locations. Accessible Grids consist of a group of mobile or fixed devices with wired or wireless connectivity and predefined or ad hoc infrastructure (Hijab & Avula, 2011). Wireless, mobile, and ad hoc grids belong to this category. In **Wireless Grids**, wireless devices can be either resource providers, which contribute to data processing and/or storage, or only be mere consumers of grid services (Furthmüller. & Waldhorst, 2010). **Mobile Grids' concept** is very similar to the Wireless Grids concept; in fact, many authors used both terms indistinguishably. (Furthmüller. & Waldhorst, 2010) define a mobile grid as a grid that includes at least a mobile device. Finally, some researchers strictly define **Ad Hoc Grids** as grid environments without fixed infrastructures, i.e., all their components are mobile (Marinescu et al., 2003).

Volunteer computing is a type of distributed computing in which people, so-called volunteers, provide computing resources to projects, which use the resources to do distributed computing and/or storage. Volunteers are usually public members in possession of their own personal computing resources (desktops, laptops, smartphones, etc.) with an Internet connection. Organizations can also act as volunteers and provide their computing resources. BOINC is an open grid framework that has been used mainly for voluntary grid computing projects (Anderson, 2004). Examples of volunteer computing projects with BOINC are SETI@home, ClimatePrediction.net, Rosetta@home, and Einstein@home.

2.2 Scheduling Strategies

Many factors make the scheduling of tasks on mobile devices challenging. Some of these factors came from their characteristics, such as battery constraints or wireless connections. The battery power is limited and supports several charging-discharging cycles, which means that resources are available for several discrete time periods (Hirsch et al., 2018). On the other hand, wireless networks make communication feasible, even while mobile devices are moving. However, wireless communications may be partially or totally interrupted at any time. Finally, mobile devices are used by people, which brings at least two important consequences when being selected as execution platforms: First, users move and can disconnect. Additionally, the devices cannot be considered dedicated platforms; hence running applications should make reasonable use of the resource.

Thus, the scheduling task has at least two purposes: the completion of tasks despite lack of battery or disconnections and making fair use of the devices' energy.

In (Curiel, 2016) scheduling strategies, in the context of mobile grids, are classified into two groups: preventive and reactive, as shown in Figure 1.

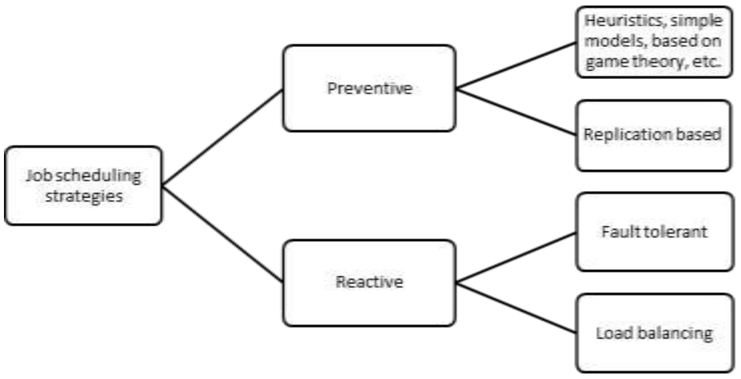
Job scheduling strategies
Preventive
Heuristics, simple models, based on game theory, etc.
Replication based
Reactive
Fault tolerant
Load balancing

Figure 1. Task scheduling strategies classification (Curiel, 2016)

Preventive scheduling tries to ensure the task culmination by selecting the most reliable resource for execution or redundant resources (replication). A suitable resource selection can be obtained through mathematical models, which may be fed by historical or current data about, for example, the status of the devices (CPU and memory utilization, CPU load, battery level, etc.) or mobility patterns, among others. On the other hand, task replication is beneficial in unreliable environments for workload with a low number of tasks and short execution times. Sometimes, however, the replication is initiated after a fault, and the strategy becomes reactive.

The use of model-based preventive strategies in mobile grids can be observed in the following works: (Vaithiya & Bhanu, 2010), (Liu & Li, 2009), (Chandak et al., 2011), (Hirsch et al., 2016), (Datta et al., 2014), (Ghosh & Das, 2010), (Birje et al., 2011), (Birje et al., 2014). Some works such as (Guo et al., 2019), (Chunlin & Layuan, 2010), and (Li & Li, 2010) focus mainly on the problem of energy limitation.

On the other hand, the work described in (Chin et al., 2009), (Litke et al., 2009), and (Du & Yu, 2010) are based on preventive replication to achieve the completion of tasks. Replication has also been used in other systems for volunteer computing. For example, the work described in (Mcgough & Forshaw, 2018) is about task replication in high throughput systems. Authors developed two forms of task replication: i) Fixed replication count, and ii) Replication counts determined through Reinforcement Learning.

In this work, we implemented 2 preventive strategies: one is based on the model proposed in (Rodriguez et al., 2010), and the other is a modification of task replication proposed in (Mcgough & Forshaw, 2018).

Reactive scheduling strategies act after the tasks have been assigned to the resources. In this case, at least two scenarios may be possible: 1) Reallocation of tasks when resources fail or are about to fail (e.g. (Lee et al., 2010) and (Adeyelu et al., 2013)) and 2) Reallocation of tasks to do load balancing among different nodes and thus maximize the number of tasks that can be completed. For example, (Rodriguez et al., 2014) explain the job-stealing technique to increase throughput by balancing the load and battery consumption.

The work described in (Hirsch et al., 2018) refers to scheduling strategies in smart mobile devices as **resource allocation (RA) mechanisms**. Authors emphasize the need to provide resource allocation mechanisms to deal with resource heterogeneity and dynamic availability that is not present in traditional distributed computing environments. The concept of **"Smart Mobile Devices (SMD) singularities**" is presented as the aspects that contribute to resource heterogeneity and dynamic availability. Three singularities are defined:

User mobility (UM): It refers to the fact that mobile device location depends on the mobility of its owner. This singularity strongly affects the quantification of mobile devices' communication resources and it can cause higher delays, error rates, and frequent spurious disconnections.

Lack of ownership (LO): this singularity refers to mobile devices' non-dedicated nature, which causes that resources such as memory or CPU time are shared with other processes and applications of the mobile device owner. Due to this non-dedicate nature of mobile devices, it is expected that external tasks not to (heavily) degrade the performance of owners' applications or experience. This problem is also present for desktop PCs of volunteer computing, e.g., BOINC.

Exhaustible Resources (ER): It is related to the fact that the energy availability of their batteries limits resources provided by smart mobile devices. This poses a new challenge to RA mechanisms since finite energy is a new heterogeneity dimension when quantifying a mobile device's resources.

Based on singularities, (Hirsch et al., 2018) also propose a classification of scheduling or RA strategies by combining UM-ER, LO-UM, and ER-LO singularities. Also, they propose two resource allocation strategies addressing independently heterogeneity derived from UM singularity and ER singularity.

BOINC-MGE component addresses all the previously mentioned singularities. Since it considers the battery power, it can be placed in the category ER-LO. Also, the modified RL REPL (Reinforcement Learning REPLication) scheduling strategy addresses the possible disconnections patterns that could come due to user mobility (UM) singularity.

2.3 BOINC

BOINC is a software infrastructure initially designed for volunteer computing, but it can also be used for grid computing (Anderson, 2004). It was originally developed at the University of California at Berkeley, and there are currently projects using BOINC in several fields such as physics, nuclear medicine, climatology, and astronomy, among others. BOINC takes advantage of a huge computing capacity, initially using personal computers worldwide, and since 2012 the processing power of mobile devices through an Android application.

A BOINC project belongs to an organization or research group interested in using the features that volunteer or distributed computing offers. The project is identified in BOINC by a master URL, which is also the website's home page. Users that want to bring their resources to a specific project uses its URL to register on it.

A project can optionally be composed of multiple applications (*Basic Concepts – BOINC*, n.d.). An application includes several programs with a specific version for different operating systems, a set of *workunits*, and *results*. A *workunit* describes the calculation to be performed. A *workunit* description has several useful parameters for the scheduling algorithm. These parameters are estimations of a) the amount of floating-point operations per second (FLOPS) that the CPU performs; b) the maximum amount of RAM that can be consumed; c) the maximum amount of disk space that can be used and, d) the total execution time.

Each calculation has results. A result consists of a reference to a *workunit* and a list of references to output files. In some cases, when task replication is used, and multiple instances of a given *workunit* are created, several results can be obtained. The use of replication in BOINC is described later.

Besides, a BOINC project has other processes that help the generation of tasks and their subsequent validation: the *work_generator* consists of a process that runs continuously on the server and is responsible for generating the *workunits* that will eventually be distributed by the scheduler to the devices. On the other hand, the *assimilator* and *validator* are responsible for verifying that the devices' results are correct and can be absorbed and marked as finalized by BOINC.

The *workunits* are sent and executed by the BOINC clients (also referred to in this paper as execution nodes or computing devices), which are the devices that volunteer users provide, being them personal computers and/or Android mobile phones.

Task Scheduling in BOINC

The task scheduling process is associated with the volunteer computing model of BOINC. When a volunteer is registered in a project, the scheduling policy executed on the client (in this case, a mobile device with Android) periodically makes requests to the task scheduler running on the server.

The task scheduler on the server responds to requests sent by computing devices. The selection of the *workunits* that will be sent to the volunteers is made through a scheduling policy based on a scoring function. The scoring function *P(N, W)* seeks to send to the execution node *N* the *workunits W* for which the value of the function *P(N,W)* is a maximum. The function used by default in BOINC (Anderson & Reed, 2009) combines parameters related to i) the expected amount of floating-point operations per second (FLOPS) of the application version available for node *N*; ii) RAM and application storage requirements; iii) the number of replicas to be sent and iv) homogeneous redundancy class to which *N* belongs (Taufer et al., 2005). The use of replicas is explained in the next section.

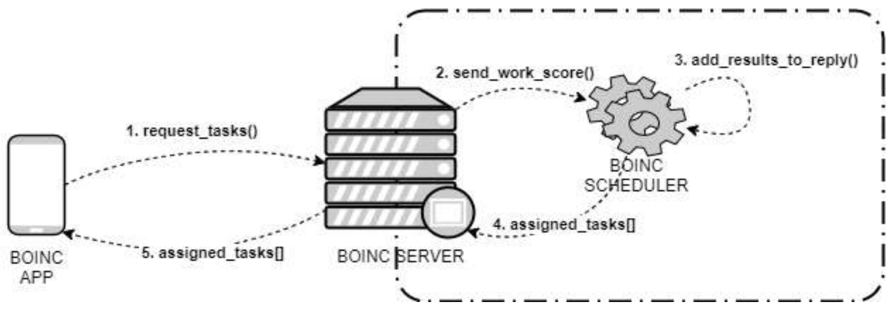

Figure 2. Task scheduling process in BOINC

Figure 2 shows the process of requesting and assigning tasks to volunteer devices in BOINC. The process starts when a client (an Android device in this case) requests tasks for execution. The BOINC SERVER received the request, which checks if the volunteer user associated with the client has enough privileges to access the project for which it is requesting tasks. After all validations and restrictions configured by the project administrator have been passed, the server passes the request to the scheduler using an internal call to the *send_work_scor*e routine, which in turn implements the scoring function described above. After selecting the tasks to be sent, the *add_results_to_reply* routine is executed to add the tasks into the *assigned_tasks* array. The group of selected tasks is returned to the BOINC SERVER where they are formatted into a response HTTP/XML message that is sent back to the client to be executed. When the tasks are finished, the client returns the result to the server and asks for new tasks, repeating all the same process. The task scheduler is implemented using the C++ programming language, and it is the component marked in the diagram as BOINC SCHEDULER. In BOINC-MGE, this component is replaced by another mechanism that integrates scheduling strategies that solve battery and disconnection challenges present in mobile grids.

Replication in BOINC

In volunteer computing, anyone with a device (smartphone, tablets, or PCs) can be a participant, and this means that it cannot be granted that all of them are reliable or well-intentioned. Additionally, since BOINC is open source and easily accessible on the Internet, there exists the risk of receiving invalid results from one or more of the execution nodes. To mitigate this risk, BOINC implements a replication mechanism in which the same *workunit* is sent to several clients. Then, through a quorum system, the server verifies that the result returned by a device matches with the results coming from other execution nodes for the same *workunit*. Each project enables this mechanism by indicating the number of replicas to be generated and the number of results required in the quorum to mark a *workunit* as completed. Thus, replication in BOINC is implemented mainly to validate the results sent by volunteers rather than increase task completion probabilities.

There is also adaptive replication, which consists of determining whether a device is reliable or not depending on the amount of consecutive satisfactory results that it returns to the server and the time it takes to execute the *workunit* compared with the execution time limit established by the system administrator. When sending a *workunit* for the first time, the server's scheduler determines whether the device is reliable. If so, no extra replicas of the *workunit* are generated. Otherwise, the number of replicas indicated by the administrator are generated. Also, the required quorum value is set equal to the number of generated replicas. Two conditions are required to mark a device as reliable: a) the device must have at least 10 consecutive executions without errors, and b) the execution time of a *workunit* does not have to exceed the average of all other tasks previously completed by the device.

This is the type of replication recommended by BOINC because it tries to reduce the number of replicas generated over time.

3. Related Work

Our research project aims to parallelly process large medical images in a distributed system consisting mainly of mobile devices. The use of mobile devices provides ubiquity and has the additional advantage that health centers can save money, space, and energy.

Instead of starting the system from scratch, we decided to evaluate the existing technologies at the beginning of the project to take advantage of the grid systems' functionalities. When evaluating the technology, one of the important requirements was the support of the C++ language since that is the language used by the ITK (Insight Toolkit) library, widely used in medical image processing.

Even though smartphones have been used for image processing and many other health-related applications, e.g. (Masciantonio & Surmanski, 2017) (Agu et al., 2013) (Rajendran, 2019), in our search of state of art, we have not found another project that addresses parallel processing of images using smartphones in a distributed architecture. The work from (Has et al., 2015) and (Kitrungrotsakul et al., 2015) describe sequential algorithms deployed in client/server architectures, where the server performs most of the work.

Regarding systems designed specifically for the use of smartphones as distributed computing platforms, we can mention the following works:

The system's goal described in (Arslan et al., 2015) is to use mobile devices at night while they are charging their battery. This project, like ours, seeks to reduce space and energy using smartphones. However, the system was developed for the execution of business applications. They use a greedy algorithm that considers the CPU and bandwidth available on each smartphone for scheduling tasks on devices. The authors also design a process migration strategy to deal with disconnections of mobile devices. Tasks assignment differs from the BOINC model: while in BOINC, volunteers must download an app to receive jobs, in the system described in (Arslan, 2014) the placement of tasks on the phones is automatically performed by the system.

CANDIS (Schildt et al., 2013) is a system similar to the one described above, designed for executing business applications. It uses a volunteer computing model where tasks are assigned to a cloud system's smartphones during working hours, while they are charging. As in our case (due to the way BOINC works), the system is not exclusive to smartphones since it allows the incorporation of desktop computers or servers. Every time a client registers at the server, it appends its computational capability and resources. Based on this data, the server can determine the size of *workunits* it sends to each client. The article also presents a simulation showing that CANDIS has the potential to reduce energy costs. CANDIS assumes a billing model where the consumer can shift energy usage to cheaper times, for example, during the night hours.

The work described in (Wagner, 2020) also implements a volunteer computing model for mobile devices' distributed computation. The system is tested with an application to identify genes associated with the development of kidney cancer. The scheduling strategy considers the phones' characteristics (e.g., slow or fast) to assign tasks. The scheduling policy is reactive concerning disconnections; when a phone is disconnected, its work is reassigned to another client.

(Kumar et al., 2019) describes a system to execute a DNA sequence similarity algorithm. The authors argue that task assignment in BOINC is complicated. Therefore they offer a simpler platform that assigns jobs automatically to volunteers and does not require any explicit changes or setups in the smartphones. Unlike our project, this paper also explores storage in mobile devices. Regarding task scheduling, the server sets deadlines for each device and ensures that the results are returned within that deadline. Tasks that fail to meet their deadlines are re-allocated to other devices.

The project described in (Salem, 2019) is a system developed to execute machine learning algorithms on a Smartphones-Based Network. The user can choose when the system can use their device and the percentage of usage. The scheduling algorithm uses this information, as well as the number of cores, threads, processes, and characteristics of the graphics card, to do the task assignments. The article does not mention strategies for disconnection cases. The application used as proof of concept is written in JavaScript.

The main difference between the previous works and ours is the type of application they were designed. These systems only support applications written in JAVA or JavaScript. In our project, BOINC allows programs written in various programming languages, particularly C++, which is one of the research project's main requirements. As in our case, systems described in (Arslan et al., 2015), (Kumar et al., 2019) and (Wagner, 2020) also consider strategies for the completion of tasks when devices are disconnected. Our project includes a simple prediction model compromised of two components that can be combined or used individually. While running on a scenario with high stability (i.e., no disconnections), the battery-aware component can be used. On the other hand, when disconnections are expected, adaptative replication can be enabled. Other works described here only face one or another scenario, not both.

4. New scheduling strategies in BOINC-MGE

As mentioned previously, we included two new scheduling algorithms in BOINC that consider the battery's current state and several scenarios of user disconnections. One of the algorithms is based on predictive models and the other on replication. One of the most important factors in selecting the strategies was their affinity to the BOINC model.

Several alternatives were considered and later evaluated using multiple criteria that weighted the easiness of implementation, the detail provided by the authors about the algorithms, the adaptability to the scheduling and deployment model of BOINC, and the evidence presented by the authors to validate their effectiveness. We focus our research on preventive strategies since implementing reactive ones would require more modifications to the current model of BOINC. Within this group, we also wanted to address the SMD singularities described before as much as possible. To address ER-LO SMD singularities, we searched and evaluated model-based strategies like the ones presented by (Rodriguez et al., 2010), (Huang et al., 2005), (Huang et al., 2006), (Li & Li, 2010), (Alenawy & Aydin, 2005), (Xie et al., 2006) and (Kim et al., 2007). Similarly, to address UM singularity, we considered replication strategies that can be used to increase task finalization probability even when one or more users disconnects. However, since we want to keep the overhead at minimal levels, we look for strategies that aim to keep the task replicas as low as possible. We evaluated in this category the strategies proposed by (Du & Yu, 2010), (Chin et al., 2009), (Mcgough & Forshaw, 2018), and the adaptive replication mechanism included in BOINC (*AdaptiveReplication – BOINC*, n.d.).

After the evaluation process SEAS (Simple Energy-Aware Scheduler) (Rodriguez et al., 2010) was selected as the strategy based on predictive models and a slightly modified version of RLREPL (Mcgough & Forshaw, 2018) as the strategy based on replication.

4.1 SEAS – Simple Energy-Aware Scheduler

As established in previous paragraphs, it is important to consider the battery conditions before selecting a mobile device to execute tasks. SEAS (Rodriguez et al., 2010) has a simple yet effective battery estimation model to achieve this goal. The model uses an algorithm that does not require to know several battery physical parameters, which are only known by manufacturers. On the other hand, the model differentiates from other complex strategies because it does not rely on the resolution of differential equations that would require the use of much computational time (Boovaragavan et al., 2008) and affect the overall system performance. Furthermore, SEAS's model uses real-time collected information about battery consumption, which means that it can predict battery discharge rate even if the mobile device is not used exclusively to execute the tasks that are sent by the BOINC server.

The SEAS algorithm consists of measuring the current battery charge (*bc*) and the current time (*ct*), and then wait until a change in the battery status charge occurs. When a change happens, the algorithm measures the new battery charge (*nbc*) and the new current time (*nct*) and uses this information to calculate the current discharge rate (*dr*) as follows:

$$dr = \frac{bc - nbc}{nct - nc} \quad (1)$$

By assuming that discharge rate is constant, the remaining time (*rt*) available for computing before battery exhaustion can be calculated as:

$$rt = \frac{nbc}{dr} \quad (2)$$

However, because the conditions of use of a device change over time (Shen et al., 2002), the discharge rate is not always constant. Therefore, SEAS proposes to keep a record of the previous estimations (*rt*) and use a historical average to determine the available time of the device. Based on these previous estimations and the execution time of previous tasks on the device, BOINC can determine the number of tasks to be sent so that they can be completed before the device is disconnected due to a lack of battery.

4.2 RL REPL – Reinforcement Learning Replication

(Mcgough & Forshaw, 2018) present the use of replication technic for an HTC (High Throughput Computing) system. Authors aim to determine for a given task *t*, submitted at time *st*, the number of extra replicas to be submitted into the HTC system. They used Reinforcement Learning (Andrew, 1998) (RL) to train an agent used to estimate the number of replicas expected to return the best reward (the chance that a task will complete within the QoS bounds).

RL is an unsupervised machine learning approach that can learn the "best" action to perform given the system's specific state. Contrary to other machine learning approaches, RL does not require big sets of historical data to train the agent since the training comes from rewards (positive feedback) given after choosing the right action and punishments (negative feedback) after choosing the wrong action, while real-time data collected from the system is used. RL adapts itself to any given environment, and due to its continuous training, RL also adapts to environmental changes.

The proposed strategy uses the reward function $R_t\ (s,a)$, obtained after executing the task *t* with several replicas at a given time *s*.

$$R_t(s,a) = \begin{cases} +k - \sigma_t & t \ completed \ within \ QoS \ bound \\ -k & t \ failed \ QoS \ bound \end{cases} \quad (3)$$

Where the first term in the reward function is used to indicate that the chosen action was good or not, and the second term (if present) helps to steer the replication task towards the minimum value; *k* is a parameter that allows quantifying how good the number of replicas selected was; *-k* indicates that the worst option for replicas was selected and +*k* is the best possible option. σ_t on the other hand, is the proportional amount of energy expended in the task's execution by the generated replicas. Moreover, $\sigma_t \in [0,k]$ quantified as a fraction of *k*, is defined as:

$$\sigma_t = \delta \min\left(1, \frac{W_t}{ad_t E}\right) \quad (4)$$

Being *a* the number of replicas, *dt* the execution time of a task *t*, *Wt* the energy wasted running task *t*, *δ ∈[0,k]* the impact we want wasted energy to have on *σt,* and *E* the average energy consumption rate for the selected resource when performing computational work.

For each task to be executed, different numbers of replicas are considered and evaluated. The options evaluated correspond to those numbers of replicas for which it has been possible to finish previous work. The one with the best reward is taken from all the possible options, thus minimizing the total amount of energy consumed.

However, reinforcement learning techniques consist of two stages: an exploratory one where new options are sought that generate better profits, and an exploitative one, which is used most of the time and aims to generate the number of replicas that offer the best reward. Exploratory choices are made by generating a random number of replicas. We do not know if an alternative that was evaluated as bad in the past suddenly became the best option due to environmental changes.

4.3 RL SEAS REPL - Reinforcement Learning SEAS Replication

The strategy implemented in this work is a modification of the RL REPL (Mcgough & Forshaw, 2018) strategy. It uses the predictions performed by the SEAS (Rodriguez et al., 2010) strategy to generate results more aligned with the state of the battery consumption of the devices connected to BOINC.

In Equation 3, the *σt* parameter is used to measure the total amount of energy wasted. We propose to rewrite it to compute the amount of energy consumed from a particular mobile node using the discharge rate as defined by the SEAS strategy multiplied by the total time taken to complete the job. Since we are using replication, we multiply this by *a*, the total number of replicas generated. Equation 3, is then defined as follows:

$$\sigma_t = \delta \min\left(1, \frac{W_t}{ad_t \frac{bc - nbc}{nct - ct}}\right) \quad (5)$$

To get the total time expended executing a task, we added some extra features in BOINC to measure every task's initial and final time of execution. Also, when summarizing the total amount of energy wasted, we use the actual data measured and reported by every mobile node.

Equation 5 is then used as the new reward function in our reinforcement learning model. When using this strategy, the BOINC-MGE scheduler checks if replication is configured and, if so, then decides whether to use an exploratory (20% of the time) or an exploitative (80% of the time) approach. When using the first one, a random number of replicas between 1 and *N* is generated; when using the exploitative approach, it uses the reward function to check which number of replicas previously generated was better considering the amount of energy wasted and the completion of tasks.

The maximum number of replicas *N* to be generated is determined and configured by the grid project administrator in BOINC-MGE. Ideally, it should be set to a number that does not exceed the actual number of mobile nodes available in the grid project. When the number of nodes is unknown (for example, because the project is open for anyone to join at any moment), some experiments should be performed, and choose a value that generates the best results.

For the 20% to 80% distribution to decide the current approach in the RL algorithm, we choose these odds as they are the ones recommended by (Mcgough & Forshaw, 2018)

The next section briefly describes how we integrate the selected new strategies into BOINC.

5. Integrating the new scheduling strategies into BOINC

To integrate the selected scheduling strategies into the BOINC platform, the server-side scheduling routine based on scoring was replaced by a new one, which in turn exposes a new API that allows the incorporation of new strategies without knowing the entire interaction performed between the scheduling routine and the BOINC-SERVER internals. The scheduling strategies were actually written on top of this new API, which requires to write only two routines, one that is used to decide which task have to be sent to the devices and another one to calculate the number of replicas that need to be generated for every task selected in the first routine. We use the SEAS algorithm in the former one and the RL SEAS REPL algorithm in the latter.

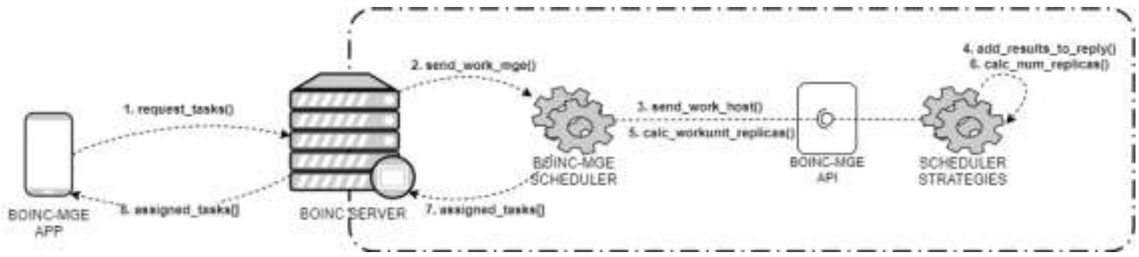

Figure 3. BOINC-MGE task scheduling process

Figure 3 shows the new task scheduling process executed in BOINC-MGE. The selected SEAS and RL SEAS REPL scheduling strategies are implemented in the **SCHEDULER STRATEGIES** component, which is invoked by BOINC-MGE through the API in the **BOINC-MGE SCHEDULER** component. At least the **send_work_host** routine must be implemented to send tasks to the clients. The SEAS strategy was implemented on this routine, and it is always called when the BOINC-MGE extension is enabled in a volunteer project. The **calc_num_replicas** routine, on the other hand, is called only if replication is enabled in the volunteer project while BOINC-MGE is also enabled. The RL SEAS REPL strategy is implemented on this routine, and it uses the battery discharge rate estimations calculated by the SEAS strategy running in **send_work_host** routine.

The Android client application (BOINC-MGE APP in Figure 3) was also updated to add the ability to collect some metrics mainly related to the battery level status and to send them to the BOINC server using the **request_tasks** routine payload message, so they can be collected and used by the SEAS strategy to do its own estimations.

6. Evaluation of BOINC-MGE scheduling strategies

In this section, we present a performance evaluation of BOINC-MGE. The evaluation is composed of two parts: Firstly, we use factorial experimental designs to evaluate the two scheduling strategies. Lastly, we present some measurements at the server-side that allow observing the new implementation's resource usage. We start the section by describing the hardware-software platform and the application used for testing.

6.1 The application

A BOINC project used to execute tasks on mobile devices was initially set up, with an application (called *canny_edge_app* as shown in Figure 4). This application segments some images by extracting its edges using the well-known Canny Edge Algorithm (Canny, 1986). To make the application suitable for a distributed project, we decided to process large image files of more than 500MB. Those files were divided into shorter images used later as the input files for the tasks generated and sent to the mobile devices.

Additionally, the application incorporates a benchmark component used to force a high CPU usage within the devices while executing the task and to increase the processing time. The application developed and adapted takes on average 12 minutes to complete every task in the mobile phones used for testing.

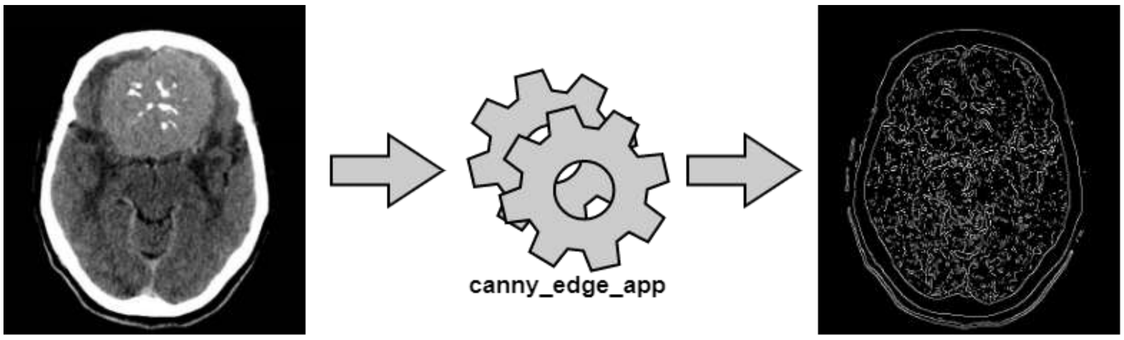

Figure 4 BOINC project application used for tests

6.2 Executing Platform

The BOINC server was installed and run in a Google Cloud virtual machine deployed in the United States using Linux Debian as the operative system, 1 vCPU (virtual CPU), and 1.8GB of RAM, and 20GB of disk space.

Five mobile devices were used in experimentation with the following characteristics:

- 4 Samsung Galaxy S4, ARM Octa-Core processor at 1.6GHz and a 2600 mAh (Milliamp Hours) battery.
- 1 Motorola Moto G, ARM Quad-Core processor at 1.2 GHz and a 2070 mAh battery.

6.3 Evaluating the model-based scheduling strategy SEAS

For the evaluation and validation of the SEAS strategy in BOINC, a $2^k r$ factorial experiment was carried out. This type of experiment allows measuring the impact that a certain number of factors have individually and jointly on one or several response variables. Each *k* factor has two possible levels, called HIGH and LOW levels. For each combination of factor and levels, a total of *r* repetitions of the experiment were performed to estimate experimental errors.

The SEAS strategy evaluation was performed using 2 factors and 3 repetitions per level ($2^2 3$), totaling 12 different experiments' executions. The response variables measured in each execution of the experiment are the total execution time (i.e., the total time spent since the scheduler sent the first sub-image until the reception of the last segment so the entire image can be reconstructed) and the average battery consumed by the devices after the completion of tasks. The objective is to evaluate the impact of the factors in each response variable. The two factors chosen were:

a) The amount of battery available in the devices. The factor's HIGH level is when the devices have enough battery to finish all the assigned work. On the contrary, the LOW level is a scenario in which some devices enter the grid with low battery.

b) The use of the SEAS scheduling strategy. Its use corresponds to the factor's HIGH level, and the LOW level is the use of the normal BOINC scheduling strategy for the delivery of the work to the devices.

6.4 Results

Table 1 shows the results obtained for each experiment's combination in the response variable “average battery consumption”. BOINC-MGE (i.e., the use of the SEAS strategy) generates less consumption than its counterpart BOINC. This is because BOINC-MGE predicts which devices will end up disconnected in the scenario with battery limitation. Based on this prediction, BOINC-MGE avoids sending them tasks that will only be partially executed, resulting in an impact on the total battery consumed throughout the grid.

In the scenario where there is no battery limit, BOINC-MGE also uses its total remaining time prediction mechanism available on the devices and sends some tasks in advance to the execution nodes. So the files needed for these tasks can be downloaded in parallel with the execution of other tasks. This allows to save the total processing time and therefore decreasing the total battery consumption.

Table 1. Average battery consumption (%) for every factor combination

	Average	Experiment 1	Experiment 2	Experiment 3	Error Exp 1	Error Exp 2	Error Exp 3
Limited battery level & BOINC	25	24	25	26	-1	0	1
Limited battery level & BOINC-MGE	20	19	21	20	-1	1	0
No battery limitation & BOINC	26	27	26	25	1	0	-1
No battery limitation & BOINC-MGE	18,67	19	19	18	0,33	0,33	-0,67

Concerning the total execution time, there is no significant difference between BOINC and BOINC-MGE when all the devices have enough battery to finish the work. However, when some of the devices have limited battery available, it becomes important to predict disconnections due to lack of battery. Based on this prediction, tasks are not delivered to devices close to disconnect, which avoids rescheduling and contributes to decreasing the total execution time. This is the SEAS strategy policy; therefore, total execution times are shorter when BOINC-MGE is used. Results can be seen in Figure 5, which shows the measurements obtained in the limited battery scenario.

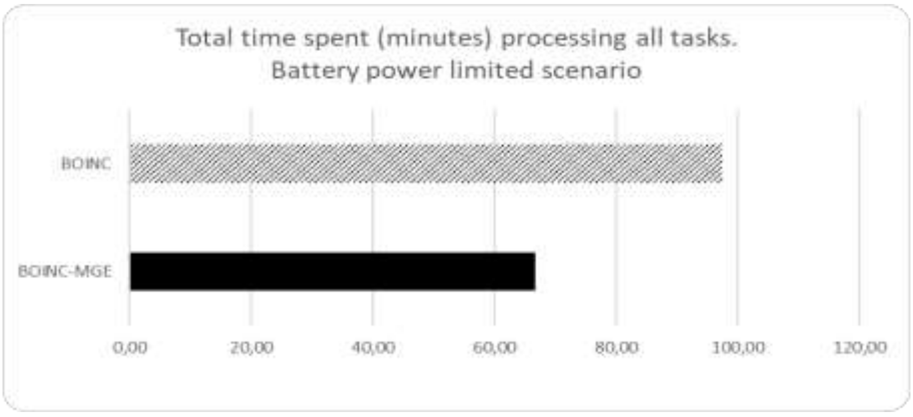

Figure 5. Total time spent processing all tasks

6.5 Evaluating the replication-based scheduling strategy ML SEAS REPL

To evaluate the effectiveness of the replicas, it is necessary to create unreliable execution scenarios. To this end, with used a two-factor full factorial experimental design. Since each factor has 4 levels, and every factor-level combination was conducted only once, the total number of experiments was 16.

The factors were:

a) The disconnection scenario, with 4 levels: the first one represents an ideal scenario where there are no disconnections. The other three scenarios have different disconnection frequencies:

   - Very unstable, with disconnections every 10 minutes.
   - Medium stability, with disconnections every 15 minutes.
   - Stable or Few disconnections, in which the devices are disconnected every 30 minutes

   These values were not randomly selected; instead, we tried to simulate 4 different scenarios by considering that the application's average execution time on any of the mobile phones is 12 minutes. Therefore, a “Very Unstable” scenario means that the mobile phone will be disconnected without completing the task assigned. “Medium Stability” means that the mobile phone would be able to complete and upload the result of at least one task before disconnecting. Similarly, “Stable” means that the node will fully complete and report at least two tasks, and the “Ideal” scenario means that the mobile phone will complete any task assigned without disconnecting.

   It is worth mentioning that the disconnection frequency is also the time a mobile device remains disconnected. Every *x* number of minutes, we disconnect several devices and reconnect others that were previously disconnected in the past.

   Since we only have 5 devices to perform our test and use from 1 to 3 replicas, we randomly select 2 devices to disconnect when our defined frequency is matched.

b) The scheduling strategy, with 4 levels, one of them corresponding to the BOINC-MGE scheduling strategy, and the other three to different BOINC fixed replication configurations, with 1, 2, and 3 replicas, respectively. It is necessary to clarify that, although BOINC replicas were not designed to ensure the completion of tasks in unreliable scenarios, they can be used within BOINC projects with this aim.

The response variables are the total execution time, and the average battery consumed by the devices after completing tasks.

6.6 Results

Figures 6 and 7 show the consolidated results grouped by scheduling strategy in each disconnection scenario. The dotted line shows the overall average measurement for the response variable. It can be observed that, for both response variables, the BOINC-MGE scheduling strategy generates better results, being in both cases below the overall average and of all the other replication options that BOINC offers. The graphs also show, as expected, that the best results are obtained when there are no device disconnections; this is since there no task rescheduling that could generate an increase in the total execution time and general battery consumption.

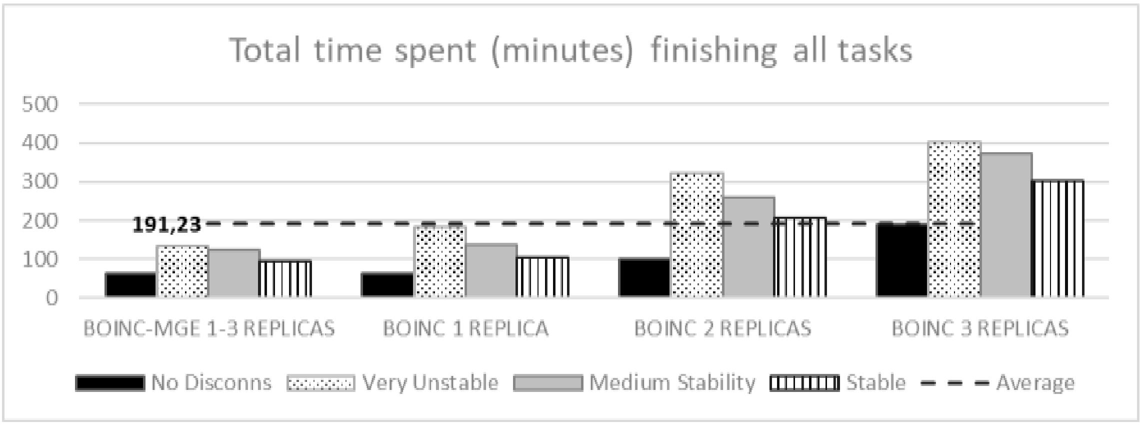

Figure 6. Total time spent (minutes) finishing all tasks

In general, using BOINC-MGE decreases on average 1 hour and 28 minutes the total time required to complete all the work dispatched and reduces the total battery consumption per device by up to 17%. These results can be explained because the RL SEAS REPL strategy can be adapted to different disconnection scenarios. It efficiently generates the number of replicas necessary to ensure the completion of tasks while reducing battery consumption.

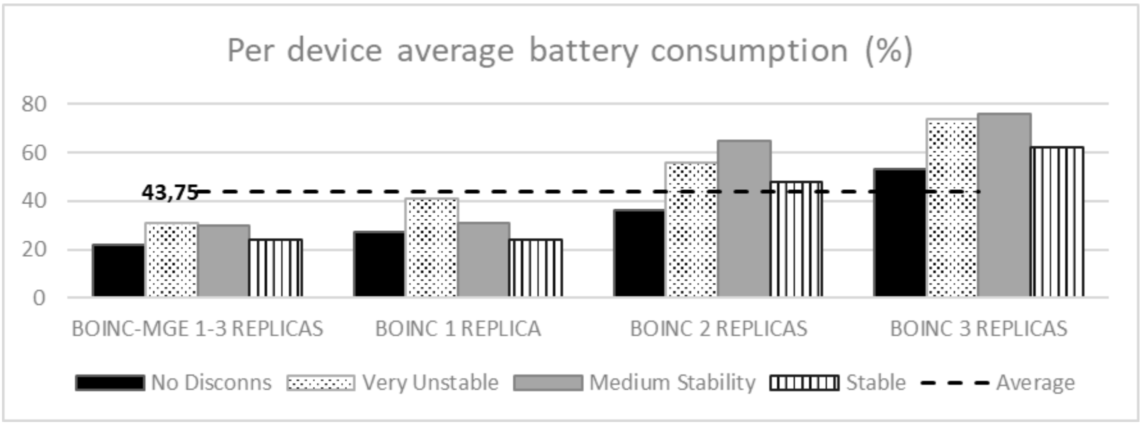

Figure 7. Per device average battery consumption (%)

6.7 Impact of BOINC-MGE in the server-side

We made measurements on the server to compare the resource usage with respect to BOINC scheduling strategies. To carry it out, it was decided to execute a set of tasks using BOINC-MGE with its two scheduling strategies enabled and the default BOINC strategy that generates a single replica for each task sent to the devices. In both cases, RAM consumption and CPU usage were measured for the entire time the devices took to finish all the tasks assigned by the scheduler.

Regarding the CPU usage, there are no substantial differences between the two alternatives, as shown in Figure 8, since BOINC-MGE does not generate new processes in the operating system nor significantly increases the calculation operations performed in the server.

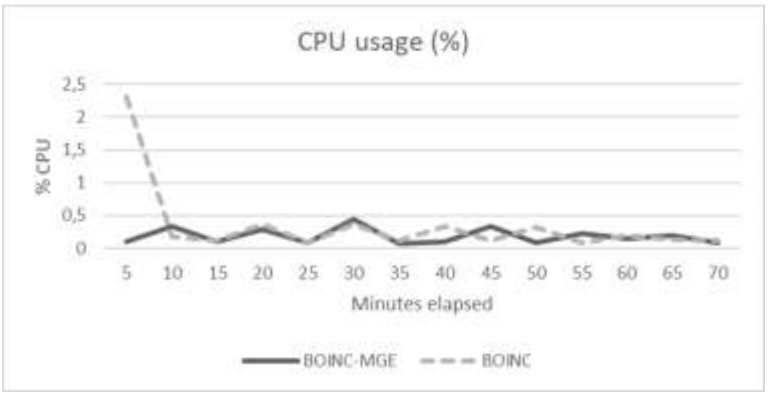

Figure 8 CPU usage (%) in the server over time

Memory usage is observed in Figure 9. Although the memory consumption is similar in both models, BOINC-MGE consumes about 1% more memory than BOINC, which is equivalent to approximately 20MBs. However, given the current hardware features that web servers have, it is not a significant consumption.

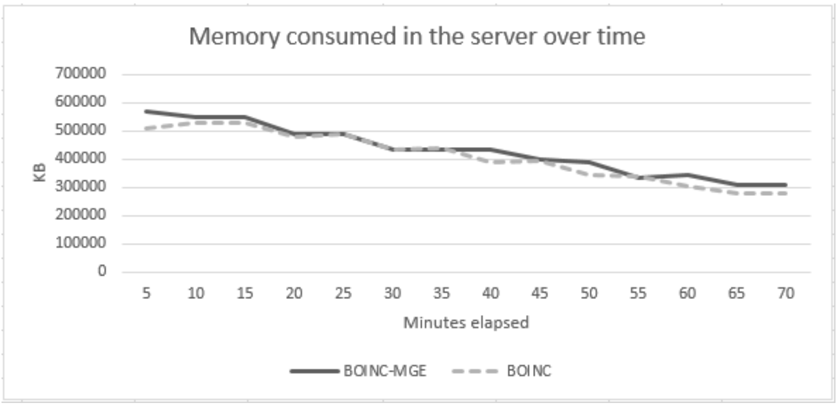

Figure 9 Memory consumed in the server over time

7. Discussion

From the obtained results, it is derived that by incorporating a simple analytical model in the scheduling strategy, it is possible to utilize the devices better. The model, parameterized with information from the devices, allowed BOINC-MGE to assign them a smaller or larger number of tasks depending on their battery status and disconnection patterns. The improvements of BOINC-MGE over BOINC were obtained in the total battery consumption and the total execution time. Nevertheless, our model could be extended for better accuracy, using other features such as network quality, network type, user behavior, etc.

To deal with disconnections, there are several strategies. One of them is the use of replication. There are preventive and reactive approaches. According to the summary presented in (Curiel, 2016), replication divides some authors' opinion: while some authors think that the strategy is right because it prevents underutilized resources, others seek new metrics that allow measuring the waste of resources due to replication. As expected, in our case study, execution times were increased by 21% on average when using replication since it was enabled when having device disconnections. Battery consumption was similar because most of the battery consumption is performed when computing tasks. In this sense, the recommendation is to use replicas only in highly unstable environments. Additionally, we recommend using adaptive strategies such as the one implemented in this work, which efficiently generates only the number of replicas necessary to ensure the completion of tasks while reducing battery consumption.

Although we focus our research on disconnections and battery exhaustion in mobile devices, there are still some other challenges related to the UM, LO singularities described before. One of them is network bandwidth and network connection type for the specific case of mobile devices. The first one refers to the quality of the network that is going to be used to send input data to the task, as well as to receive the results after the task completion. Another challenge refers to the fact that mobile devices can be connected to the internet, whether using a cellular data network or a Wi-Fi one. In order to ensure the completion of jobs and reduce the impact on the user experience and resources. An efficient mobile grid scheduler should also consider these two factors in the allocation and distribution routine.

Some articles addressing the subject are (Quintin & Wagner, 2012), (Lee et al., 2013), (Jiang et al., 2016) and (Guo et al., 2019). However, they only focus on the use of network bandwidth to decide how to distribute tasks on the grid, and they do not integrate them with other strategies to make a fully capable mobile grid scheduler that aims to solve more mobile phone singularities.

We propose exploring these areas for future work and try to incorporate more features in the scheduler routines. Nevertheless, it is worth mentioning that BOINC and BOINC-MGE currently provide a mechanism in the Android application configuration to let the user decide whether to join a specific volunteer project using mobile cellular data connections. We think that this is not enough and can be highly improved.

8. Conclusions

In this paper, we have presented BOINC-MGE, an extension to BOINC that adds new scheduling algorithms focused on solving problems related to the execution of tasks on mobile devices. These algorithms allow an adequate selection of mobile devices to complete the tasks while considering the current battery charge level and several scenarios of user disconnections.

We incorporated two state of the art algorithms, one based on a prediction model and another based on replications. An API was designed to make future incorporations of new scheduling strategies easier.

Additionally, a performance evaluation study was carried out through the execution of factorial experimental designs. Through the experiments, we were able to verify that the use of a simple analytical model (the SEAS model) parameterized with the state of the battery allows making more efficient work assignments. The general battery consumption was improved, as well as the execution times. Concerning the total execution times, we did not obtain a significant difference between BOINC and BOINC-MGE when all the devices have enough battery to finish the work. However, when some of the devices have limited battery available, the prediction of possible disconnections was crucial. Based on this prediction, tasks were not delivered to devices close to disconnect, thus avoiding rescheduling and contributing to the decrease of the total execution time.

In general, the use of replicas and the existence of intermittent scenarios, as expected, increase the execution times compared to an ideal scenario with no disconnections. However, in those cases, when using BOINC-MGE, the execution times decrease 1 hour 28 minutes on average, compared to BOINC. The total battery consumption per device was reduced by up to 17%. In the face of possible disconnection scenarios that require the use of replicas, the adaptive algorithm RL SEAS REPL only generates the number of replicas necessary to ensure the completion of tasks. Thus, if the context demands replicas, the best alternative is the use of an adaptive strategy that guarantees a more efficient use of resources.

Our results reinforce the fact that it is important to adapt scheduling strategies to mobile devices' characteristics when thinking about them as execution platforms. This will allow for better application performance and better use of the resources offered by volunteers. The above leads us to recommend the activation of BOINC-MGE in projects that incorporate mobile devices as computational resources.

The implemented models can still be refined to consider, for example, the characteristics of the interconnection network or user preferences.

In the context of our research project, some questions should still be solved. For example, what specific kind of medical imaging processing problems can it be used for? Is it convenient to continue with the volunteer computing model that BOINC has? How can image processing libraries be integrated into BOINC-MGE?

References

AdaptiveReplication – BOINC. (n.d.). Retrieved April 30, 2018, from https://boinc.berkeley.edu/trac/wiki/AdaptiveReplication

Adeyelu, A., Olajubu, E., Aderounmu, A., & Ge, T. (2013). A Model for Coordinating Jobs on Mobile Wireless Computational Grids. *International Journal of Computer Applications*, *84*(13), 17–24. https://doi.org/10.5120/14636-1637

Agu, E., Pedersen, P., Strong, D., Tulu, B., He, Q., Wang, L., & Li, Y. (2013). The smartphone as a medical device: Assessing enablers, benefits and challenges. *2013 IEEE International Workshop of Internet-of-Things Networking and Control, IoT-NC 2013*. https://doi.org/10.1109/IoT-NC.2013.6694053

Alenawy, T. A., & Aydin, H. (2005). Energy-constrained scheduling for weakly-hard real-time systems. *Proceedings - Real-Time Systems Symposium*.

https://doi.org/10.1109/RTSS.2005.18

Anderson, D. P. (2004). BOINC: A system for public-resource computing and storage. *Proceedings - IEEE/ACM International Workshop on Grid Computing*, 4–10. https://doi.org/10.1109/GRID.2004.14

Anderson, D. P., & Reed, K. (2009). Celebrating diversity in volunteer computing. *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS*, 1–8. https://doi.org/10.1109/HICSS.2009.105.

Andrew, A. M. (1998). Reinforcement Learning: An Introduction. In *Kybernetes*. https://doi.org/10.1108/k.1998.27.9.1093.3

Arslan, M. Y., Singh, I., Singh, S., Madhyastha, H. V., Sundaresan, K., & Krishnamurthy, S. V. (2015). CWC: A distributed computing infrastructure using smartphones. *IEEE Transactions on Mobile Computing*. https://doi.org/10.1109/TMC.2014.2362753

Basic Concepts – BOINC. (n.d.). Retrieved April 30, 2018, from https://boinc.berkeley.edu/trac/wiki/BasicConcepts

Birje, M. N., Manvi, S. S., & Das, S. K. (2014). Reliable resources brokering scheme in wireless grids based on non-cooperative bargaining game. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2013.07.007

Birje, Manvi, & Bulla. (2011). Economical job scheduling in wireless grid. *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology*. https://doi.org/10.1109/ICECTECH.2011.5941835

Boovaragavan, V., Harinipriya, S., & Subramanian, V. R. (2008). Towards real-time (milliseconds) parameter estimation of lithium-ion batteries using reformulated physics-based models. *Journal of Power Sources*. https://doi.org/10.1016/j.jpowsour.2008.04.077

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.1986.4767851

Chandak, A., Sahoo, B., & Turuk, A. K. (2011). Efficient task scheduling using mobile grid. *International Conference on Recent Trends in Information Technology, ICRTIT 2011*, 1255–1258. https://doi.org/10.1109/ICRTIT.2011.5972343

Chin, S. H., Suh, T., & Yu, H. C. (2009). Genetic algorithm based scheduling method for efficiency and reliability in mobile grid. *Proceedings of the 4th International Conference on Ubiquitous Information Technologies and Applications, ICUT 2009*, 1–6. https://doi.org/10.1109/ICUT.2009.5405741

Chunlin, L., & Layuan, L. (2010). Controlling energy without compromising system performance in mobile grid environments. *Computers and Electrical Engineering*, *36*(3), 503–517. https://doi.org/10.1016/j.compeleceng.2009.12.004

Curiel, M. J. H. (2016). Wireless grids: Recent advances in resource and job management. *Handbook of Research on Next Generation Mobile Communication Systems*, 293–320. https://doi.org/10.4018/978-1-4666-8732-5.ch012

Datta, P., Dey, S., Paul, H. S., & Mukherjee, A. (2014). ANGELS: A framework for mobile grids. *Proceedings - International Conference on 2014 Applications and Innovations in Mobile Computing, AIMoC 2014*, 15–20. https://doi.org/10.1109/AIMOC.2014.6785513

Du, L. J., & Yu, Z. W. (2010). Scheduling algorithm with respect to resource intermittence in mobile grid. *2010 6th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2010*, 2–6. https://doi.org/10.1109/WICOM.2010.5600181

Duan, L., Kubo, T., Sugiyama, K., Huang, J., Hasegawa, T., & Walrand, J. (2014). Motivating smartphone collaboration in data acquisition and distributed computing. *IEEE Transactions on Mobile Computing*, *13*(10), 2320–2333. https://doi.org/10.1109/TMC.2014.2307327

Furthmüller., & Waldhorst. (2010). A survey on grid computing on mobile consumer devices. *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing*, 313–363. https://doi.org/10.4018/978-1-4666-0879-5.ch510

Ghosh, P., & Das, S. K. (2010). Mobility-aware cost-efficient job scheduling for single-class grid jobs in a generic mobile grid architecture. *Future Generation Computer Systems*, *26*, 1356–1367. https://doi.org/10.1016/j.future.2009.05.003

Guo, S., Liu, J., Yang, Y., Xiao, B., & Li, Z. (2019). Energy-Efficient Dynamic Computation Offloading and Cooperative Task Scheduling in Mobile Cloud Computing. *IEEE Transactions on Mobile Computing*, *18*(2), 319–333. https://doi.org/10.1109/TMC.2018.2831230

Has, M., Kaplan, A. B., & Dizdaroğlu, B. (2015). Medical image segmentation with active contour model: smartphone application based on client-server communication. *2015 Medical Technologies National Conference* (TIPTEKNO), 1-4.

Hijab, M., & Avula, D. (2011). Resource discovery in wireless, mobile and ad hoc grids - Issues and challenges. *International Conference on Advanced Communication Technology, ICACT*, 502–505.

Hirsch, M., Mateos, C., & Zunino, A. (2018). Augmenting computing capabilities at the edge by jointly exploiting mobile devices: A survey. *Future Generation Computer Systems*, *88*, 644–662. https://doi.org/10.1016/j.future.2018.06.005

Hirsch, M., Rodríguez, J. M., Mateos, C., & Zunino, A. (2017). A Two-Phase Energy-Aware Scheduling Approach for CPU-Intensive Jobs in Mobile Grids. *Journal of Grid Computing*, *15*(1), 55–80. https://doi.org/10.1007/s10723-016-9387-6

Hirsch, M., Rodriguez, J. M., Zunino, A., & Mateos, C. (2016). Battery-aware centralized schedulers for CPU-bound jobs in mobile Grids. *Pervasive and Mobile Computing*, *29*, 73–94. https://doi.org/10.1016/j.pmcj.2015.08.003

Huang, C. Q., Zhu, Z. T., Wu, Y. H., & Xiao, Z. H. (2006). Power-aware hierarchical scheduling with respect to resource intermittence in wireless grids. *Proceedings of the 2006 International Conference on Machine Learning and Cybernetics*, *2006*(August), 693–698. https://doi.org/10.1109/ICMLC.2006.258419

Huang, Mohapatra, S., & Venkatasubramanian, N. (2005). An energy-efficient middleware for supporting multimedia services in mobile grid environments. *ITCC 2005: International Conference on Information Technology: Coding and Computing, Vol 2*, 220–225. https://doi.org/10.1109/itcc.2005.77

Jiang, J., Ma, S., Li, B., & Li, B. (2016). *Symbiosis : Network-Aware Task Scheduling in Data-Parallel Frameworks*. In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. 1-9.

Kim, K. H., Buyya, R., & Kim, J. (2007). Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. *Proceedings - Seventh IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2007*, 541–548. https://doi.org/10.1109/CCGRID.2007.85

Kitrungrotsakul, T., Dong, C., Tateyama, T., Han, X.-H., & Chen, Y.-W. (2015). Interactive segmentation and visualization system for medical images on mobile devices. *Journal of Advanced Simulation in Science and Engineering*, *2*(1), 96–107. https://doi.org/10.15748/jasse.2.96

Kumar, M., Bhat, R. R., Alavandar, S. R., & Ananthanarayana, V. S. (2019). Distributed Public Computing and Storage using Mobile Devices. *2018 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics, DISCOVER 2018 - Proceedings*, 82–87. https://doi.org/10.1109/DISCOVER.2018.8674111

Kurdi, H., Li, M., & Al-Raweshidy, H. (2008). *A Classification of Emerging and Traditional Grid Systems*. IEEE Distributed Systems Online. *9*(3).

Lee, Abe, H., Hirotsu, T., & Umemura, K. (2013). Performance implications of task scheduling by predicting network throughput on the internet. *Proceedings - 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013*, 1089–1098. https://doi.org/10.1109/TrustCom.2013.132

Lee, Chin, S. H., & Gil, J. M. (2010). Efficient resource management and task migration in mobile grid environments. *Communications in Computer and Information Science*, *78 CCIS*, 384–393. https://doi.org/10.1007/978-3-642-16444-6_48

Li, & Li. (2010). Energy constrained resource allocation optimization for mobile grids. *Journal of Parallel and Distributed Computing*, *70*(3), 245–258. https://doi.org/10.1016/j.jpdc.2009.06.003

Litke, A., Halkos, D., Tserpes, K., Kyriazis, D., & Varvarigou, T. (2009). Fault tolerant and prioritized scheduling in OGSA-based mobile Grids. *Concurrency Computation Practice and Experience*. https://doi.org/10.1002/cpe.1351

Liu, L., & Li, C. (2009). Mobile grid task scheduling considering resource reliability. *Proceedings - 1st International Symposium on Computer Network and Multimedia Technology, CNMT 2009*. https://doi.org/10.1109/CNMT.2009.5374742

Marinescu, D. C., Marinescu, G. M., Ji, Y., Bölöni, L., & Siegel, H. J. (2003). Ad hoc grids: Communication and computing in a power constrained environment. *IEEE International Performance, Computing and Communications Conference, Proceedings*, 113–122. https://doi.org/10.1109/pccc.2003.1203690

Masciantonio, M. G., & Surmanski, A. A. (2017). Medical smartphone applications A new and innovative way to manage health conditions from the palm of your hand. University of Western Ontario Medical Journal. 86(2), 51-53, 2017.

Mcgough, A. S., & Forshaw, M. (2018). Evaluation of Energy Consumption of Replicated Tasks in a Volunteer Computing Environment. *In Companion of the 2018 ACM/SPEC International Conference on Performance Engineering* . 85–90.

Mengistu, T. M., & Che, D. (2019). Survey and taxonomy of volunteer computing. *ACM Computing Surveys*, *52*(3). https://doi.org/10.1145/3320073

Quintin, J. N., & Wagner, F. (2012). WSCOM: Online task scheduling with data transfers. *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, 344–351. https://doi.org/10.1109/CCGrid.2012.21

Rajendran. (2019). Image Analysis Using Smartphones for Medical Applications: A Survey. Intelligent Pervasive Computing Systems for Smarter Healthcare. 275-290.

Rodriguez, J. M., Mateos, C., & Zunino, A. (2014). Energy-efficient job stealing for CPU-intensive processing in mobile devices. In *Computing*, 96(2). https://doi.org/10.1007/s00607-012-0245-5

Rodriguez, J. M., Zunino, A., & Campo, M. (2010). *Mobile Grid SEAS : Simple Energy-Aware Scheduler*. *54*(2293), 3341–3354.

Salem, H. (2019). Distributed Computing System on a Smartphones-Based Network. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11771 LNCS*(November), 313–325. https://doi.org/10.1007/978-3-030-29852-4_26

Schildt, S., Büsching, F., Jörns, E., & Wolf, L. (2013). CANDIS: Heterogenous mobile cloud framework and energy cost-aware scheduling. *Proceedings - 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-IThings-CPSCom 2013*, 1986–1991. https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.372

Shen, W. X., Chan, C. C., Lo, E. W. C., & Chau, K. T. (2002). Estimation of battery available capacity under variable discharge currents. *Journal of Power Sources*. https://doi.org/10.1016/S0378-7753(01)00840-0

Taufer, M., Anderson, D., Cicotti, P., & Brooks, C. L. (2005). Homogeneous redundancy: A technique to ensure integrity of molecular simulation results using public computing. *Proceedings - 19th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2005*, *2005*. https://doi.org/10.1109/IPDPS.2005.247

Vaithiya, S. S., & Bhanu, S. M. S. (2010). Scheduling tasks in mobile grid environment using mobility based resource prediction. *2010 1st International Conference on Parallel, Distributed and Grid Computing, PDGC - 2010*, 89–94. https://doi.org/10.1109/PDGC.2010.5679600

Wagner, J. (2020). A Prototype for Distributed Computing Platform. *Technical Library*.

350. https://scholarworks.gvsu.edu/cistechlib/350

Xie, T., Qin, X., & Nijim, M. (2006). Solving energy-latency dilemma: Task allocation for parallel applications in heterogeneous embedded systems. *Proceedings of the International Conference on Parallel Processing*, 12–19. https://doi.org/10.1109/ICPP.2006.66