Machine Learning Assisted Heuristic Approach for Optimal Task Deployment in Hybrid Cloud/Edge Environments

Timothy Lynar*, Fatemeh Jalali, Ramachandra Rao Kolluri, Olivia J Smith and Frank Suits
IBM Research Australia, Victoria, Australia

Abstract

Dynamically allocating tasks in an IoT environment can benefit from resource state prediction, task requirements and various forms of machine learning. This paper proposes a distributed task allocation scheme, for IoT networks in a hybrid Edge-Cloud platform, which relies on a combination of machine learning and heuristic methods in an auction-based system with an objective to provide near-optimal task allocation. The proposal is tested on several workloads, developed using real-time data from IoT and Cloud computing environments, and the results are compared against various simple-heuristic and complex-optimization-based allocation schemes. The results show that the proposal approaches a near-optimal allocation of tasks to resources with low decision-making overhead owing to the lightweight learning system.

1 Introduction and related work

Applications or tasks executed on IoT devices are typically allocated to either the device that initiates the application or offloaded to a predefined Cloud computing resource in a static manner. However, there are wide-spread investigations into using *hybrid cloud/edge* computing architecture to offload tasks to compute resources in an IoT device environment. It has been observed that fixed selection of platforms during runtime can significantly reduce the performance of mobile IoT applications. Several works [1; 2; 3] have proposed dynamic selection methods that modify platform allocation during runtime.

Edge-computing platforms emerged to serve the IoT applications that cannot be served by the *cloud* due to a plethora of network and computational constraints. Edge-computing, in general, takes advantage of local computation and storage to perform most of the tasks in an IoT workload locally and only send remaining tasks or information to the cloud. Hybrid cloud/edge is another platform which has been recently defined to further fine tune the task deployment based on the computational intensity of the tasks. For example, in a hybrid edge/cloud environment simple tasks like feature extraction, data smoothing etc. are carried out locally and heavy computation tasks like complicated machine learning algorithms, predictive analytics are transferred to the cloud.

As mobile computing became more pervasive, computational offloading has increasingly been a focus of research. In this paper we are designing solutions that are able to overcome the limitations imposed by underpowered processors, lack of memory resources, and energy constraints that were typically present in low power IoT devices. Majority of the IoT resource allocation systems found in the literature perform tasks such as data transmission and protocol translation with limited embedded intelligence. There is constrained ability to dynamically allocate tasks between platforms based on real-time conditions [4; 5]. However, for any given task its allocation to a platform will impact not just the execution speed of that task but the IoT environment as a whole. If it were executed on a platform exterior to the IoT environment, such as the Cloud, it may in turn conserve the otherwise used resources within the IoT environment such as compute capacity, energy etc.

Phone2Cloud [6] is one example that dynamically selects when to offload applications, but requires custom applications. Other work such as ThinkAir [1], MAUI [2], Clone Cloud [7], and Jade [8] make dynamic decisions on offloading an application at runtime. n more detail, the authors of [1] introduce a framework to offload mobile computation to the Cloud. This framework profiles an application's characteristics when executing on a mobile phone and also when executing on Cloud compute resources. Additionally, the framework monitors network activity and mobile phone device status at runtime. This framework focuses on the elasticity and scalability of the Cloud and parallel execution of offloaded applications using multiple virtual machine (VM) images. On the other hand, [2] proposes a framework that optimises energy consumption of a mobile device by offloading computationally intensive applications dynamically to a remote resource. The framework considers the trade-off between the energy consumption of local computation versus data transmission energy consumption for remote execution. This framework only focuses on energy usage and it does not consider other metrics such as application performance or the

*Corresponding author email: timlynar@au.ibm.com.
T. Lynar, F. Jalali, R. R. Kolluri, O. J. Smith and F. Suits are with IBM Research – Australia, 22/60 City Road, Southbank VIC 3006 Australia.

cost of delaying execution for offloading both data and computation. The authors of [7] presents a framework for improving the battery life and performance on mobile devices through offloading computationally intensive components to the Cloud using a cloned virtual machine (VM) image. This framework uses an off-line static analysis of various scenarios with different conditions on both mobile devices and the Cloud. The result of this analysis is stored in a database to enable identification of application components to offload. This solution is limited by requiring the database be updated manually as new applications are added to the system. One approach that has been explored is the creation of frameworks that allow programmers to statically annotate the components of an application and associated data that can be offloaded, thus assisting in the execution of computationally expensive applications whilst minimising the impact on battery life.

In the majority of the literature, application's performance, the real-time conditions of the network, and local hardware utilisation are monitored. The decision on where to execute an application is made in real time according to the application's attributes and the availability of resources. However, some of these works only consider mobile phone and cloud based networks, present limited working examples, and make use of parallelization on virtual machines to speed up the applications. A few papers in this domain, for example, [9] do not clearly specify whether application offloading is static or dynamic. The idea of offloading computation to remote resources has been studied in depth [1], [10] and [7] but this current solution is different because it is specific to the IoT domain and leverage machine learning techniques to learn and predict task performance in an environment of dynamic resource capabilities and also use platform independent deployment capability.

1.1 Contribution

In an IoT environment, resource providers and resource consumers may be one and the same, co-located, or on opposite sides of the world. They may have different sensors, network connections, and processing performance. Additionally, devices may be network, compute, energy or power constrained, while tasks produced by resource consumers may be frequent and predictable or unpredictable and event-driven. However, the use of dynamic distributed task allocation through the use of machine learning is yet to be fully explored to the best of our knowledge.

The contribution of this paper is two-fold. Firstly, we design an IoT workflow manager that acts as a distributed resource allocation system and strives to allocate tasks to compute resources, be they on the cloud, local, or on the edge within an IoT environment in a manner that optimally meets the requirements and constraints of both resource consumers and resource providers. Our IoT workflow manager that uses the described machine learning heuristic (MLAH) approach can examine a set of tasks for their execution requirements, e.g. network, computational and latency, and optimally allocate those tasks to the most desirable match of resources, whether those resources are local, Cloud or Edge. The ability to perform this match in a distributed fashion is obtained by leveraging machine learning techniques to forecast a given application's performance on a given resource based on that resource's prior execution history, current computational resources. In another variant of the proposal, the resource providers act as the learning nodes leading to a distributed learning based allocation system. The goals of the IoT workflow manager are to simultaneously allow both resource consumers and resource providers to negotiate numerous (and sometimes conflicting) objectives. Secondly, we examine our resource allocation approaches on a number of real-time workload based on data obtained from real IoT and Cloud computing environments. The results are compared to several centralized, distributed and random allocation methods.

1.2 Organization

This paper is organized as follows: The task allocation problem and relevant state of the art is explored and brought to context in Section 1. Section 2 introduces the main proposal of the paper, the IoT workflow manager. This section also describes the architecture that consists of various modules/subsystems working in cohesion to enable intelligent allocation of tasks to IoT nodes based on an auction-based mechanism. Section 3 explain the data optimization benchmark and reports the simulation results. Finally, a summary of the proposed methods and related future work is presented in Section 4.

2 IoT workflow manager

The IoT workflow manager described in this paper builds on insights obtained from the aforementioned prior work. As in the prior work we describe a system that monitors the execution of applications and consider the characteristics of available hardware, network resources and the applications expected execution requirements to make dynamic resource allocation decisions. In addition, our proposal is completely distributed and brings resource allocation intelligence to the system through machine learning, to learn the application and environment characteristics over time in order to make decisions dynamically for saving energy, and reducing execution time of IoT applications.

2.1 Architecture

The IoT workflow manager is a distributed task resource allocation system with no centralised components. Broadly, it consists of two resource allocation decision making components: an estimator and an auctioneer. The estimator estimates the performance of a task of known characteristics on any known node with limited history. The auctioneer uses economic resource allocation techniques to reconcile competing objectives. These components work in unison to produce a pseudo-optimal outcome for the system as a whole. The architecture consists of three classes of resources:

- **resource providers** - which are nodes that can compute tasks;
- **resource consumer/requesters** - which are nodes that request tasks be executed;
- **service nodes** - which host services such as a data store or an image registry.

Any node may have one or many of the aforementioned roles within the system. Other components of the system include networking infrastructure, and image repositories. The IoT workflow manager may include a multitude of gateway devices. An IoT Gateway device is one which provides access between nodes connected to it the nodes connected to related gateways via a virtual private network (VPN). In this way the gateway bridges the communication between two or more remote sites.s

2.2 Client server interaction

Each resource provider acts as a server that responds to various commands through a restful API. The resource providers can be queried by resource consumers on their status, execution history, and their bid for a new task. Each resource provider may also act as resource consumer, as may nodes that are not resource providers. Resource consumers request bids, process those bids based on their own objectives, and request the execution of tasks.

When a task is submitted, the resource consumer, which is the node submitting the task, requests a bid. Resource providers, generally including the node that is submitting the task, calculate their bid. They then return their bid to the resource consumer. The resource consumer then builds a model to predict the execution time of this task on each resource that has returned a bid. To build that model, the resource consumer requests an execution history update from each node that has returned a bid.

Fit and predict functions are then executed to predict the execution time of the task on each node that has returned a bid. The predicted execution time is then weighted based on the bid returned by each node. This weighted bid enables the resource allocator to differentiate between otherwise homogeneous nodes, based upon their current state. The bids are sorted and the largest bid is taken. This will be called the bid winner. A request is then made to the bid winner to queue and download the required data, if any is required.

The bid winner then downloads that data. A request is then made to the bid winner to queue the execution of the task. All tasks are docker-based, therefore each execution will require the acquisition of the required container. If the container already exists on the bid winner then no new download will occur, otherwise the bid winner will connect to the image registry and download the required container. The task will then be executed.

2.3 Estimator

The estimator is where the machine learning component is utilized. Using historical run statistics, it builds a regression model $g(\cdot)$ on the execution command (the specific task) cmd, the node data $node$, and the bid of each node, $node_{bid}$. Subsequently the estimator forms a linear model based on the data provided. Using the model, the estimator derives a performance modified bid value for the given task and evaluates the best bid under offer.

Model of estimated time to complete a task
$time \sim g(cmd, node, node_{bid})$

This model is then applied to predict the execution time of new task on any known node. This model benefits from the exchange of history with other nodes.

2.4 Estimators on resource providers

Performing the machine learning on a per bidder basis will become computationally expensive in high volume environments as such we have also explored an alternative method, for use when running at scale. The bidding function f method, which will scale, but will require history to learn its own performance. As such it is suitable only after the task being executed has been executed a number of times on the bidding node. As soon as a resource provider receives a request to bid for a particular type of task, it computes a bidding function f_{bid} (linear regression) of historical governor state, i.e. compute time versus an input parameter of request type (for example, size of the picture in a face detection task, etc.) and sends it to the resource consumer. The requester, on receiving bidding functions from various bidders can use them to assess the best bid for the requested task.

Based on task allocation, the winning bidders can develop improved bidding functions using the new input data and compute times. Machine learning based bidding functions $f_{bid,ML}$ will benefit the task allocation paradigm. Such a system will improve the bidding functions of resource providers over time. For a series of tasks or much more complex workloads.

2.5 Auctioneer

The auctioneer is the component that requests and sorts bids from nodes. This component always sits on the resource consumer. In the terminology of auctions, the bids are single shot, first price, and sealed; that is, each resource provider bids only once for a given task, and do not see each other's bids. To enable the process to scale, bids are considered to be persistent between bid requests. Each resource provider's bid is based upon known data of its performance and limitations and defined by its limiting factor, described as its governor. The three governors we currently consider are speed, energy, and network, however additional governors may be defined. The bid can be viewed as the resources status. If the resource provider is an IoT device that is network-constrained, then the network governor may be selected; if the node is energy constrained then the energy governor selected. Otherwise if the node is compute constrained then speed will be selected. All governors are currently designed to include a common element of accounting for the current computational usage of a given node. Each governor is described by the method of calculating bids. The logic used by each governor to calculate their bid is given in Table 1. In Table 1, c is the CPU usage as a percentage, r is the ram usage as a percentage, p is the ping to resource requester as percentage of maximum acceptable ping and $e_{rate} = s_{battery}/7200$ with $s_{battery}$ as the seconds of battery available on the IoT node.

2.6 Auction process

The auction occurs on the resource requester when the resource requester requests a new resource. The bids come from the resource providers but are modified by the resource

Governor logic	Bid
Speed	$1 + ((-c - r)/200)$
Energy	$1 + (((-c - r) * e_{rate})/200)$
Network	$1 + ((-c - r - p)/300)$

Table 1: Bid calculation for various governors.

providers governor (speed, energy or network). The bids do not tell the resource requester anything about the performance of the node, it provides an indication of the nodes current load and limitations and as such the resource requester could infer likely relative performance of the node relative to that nodes prior performance. It can be thought of as how much the resource wants to be utilised at any given point in time. The machine learning decision making component is complementary to the bidding process, it predicts the likely execution time of a given task on all known nodes.

2.7 Remarks

Remark 1. *For any task the wall-time is defined as:*

$$t_{wall} = t_{conn} + t_{udata} + t_{dn} + t_{ddata} \quad (1)$$

where

- t_{conn} *is the time taken to connect to the best bidder node,*
- t_{udata} *is the time taken to upload the required data to the node - network bandwidth dependent,*
- t_{dn} *is the time taken by the bidder to compute the given data as in Table 5 , and*
- t_{ddata} *is the time taken to download the results form the compute node - network bandwidth dependent.*

We take into account that the connection, upload/download times are smaller than the task computation times for the tasks allocate, i.e., $t_{dn} \gg t_{conn} + t_{udata} + t_{ddata}$*, so the computation times have the largest effect on the wall-time in such a model. However, there may be scenarios where the computation times are lesser than or equal to the connection, data upload/download times, in which case they also have to be considered for optimal allocation of resources in the IoT network. While there are many parameters to optimize in allocating resources in an IoT network, we have chosen the computation time or wall-time as the key parameter for representation. The proposed method is adaptable to any (or a weighted sum of) parameter/s such as computation power, energy efficiency, etc.*

Remark 2. *The IoT network considered in this work is static and the connections are not driven/affected by the movement of IoT nodes. Allocating network resources using the proposed MLAH approach for dynamic/non-stationary networks is an interesting dynamic control problem and is left for future research.*

Remark 3. *The estimators that are bidding functions for a task, as seen in section 2.4, begin with no data or zero bidding functions for the given task. Therefore the auctioneer will select a random zero bid until all the resource providers develop some bidding functions. Once the bidding functions are developed the estimator on each resource provider will continually work on improving the bidding functions leading to transparent and efficient bidding process.*

3 Results and discussion

3.1 Data

In order to examine the performance of the workflow manager we have recorded the values and characteristics from a number of real devices. In this section we explain how those values are converted into bids. Three real devices have been examined, two are Cloud instances and one is an ARM based IoT device, typical of devices used in the IoT space for prototyping (see Table 2). The mean execution time from ten runs of the test command are reported in Table 3.The test command used was the sysbench benchmark [11]. All values for node MIPS, RAM, and CPU characteristics were discovered using user-land tools and interrogating the kernel on each device. Eight workloads have been examined. Each workload is run on the three aforementioned nodes (see Table 2). Each workload is made up of uniform tasks of the type defined above, the time to compute a task on each node is shown in Table 3. A description of each workload is provided in Table 4.

Table 2: Attributes of nodes used in the theoretical examination of the speed governor

Name	Type	Num cores	Bogomips MiB	RAM	CPU
Node1	Cloud	16	4000.14	16047	x86 2GHz
Node2	Cloud	1	5200.16	1992	x86 2.6GHz
Node3	IoT	4	38.4	859	ARMv7

Table 3: Mean test results

Name	Mean run time seconds
Node1	0.093
Node2	0.085
Node3	1.753

Table 4: Description of all test workflows showing total tasks and task distribution over time

Test	Total tasks	Tasks distribution
1	12	12 per second
2	24	12 per second
3	48	12 per second
4	96	12 per second
5	12	All at step 0
6	24	All at step 0
7	48	All at step 0
8	96	All at step 0

3.2 Optimization benchmarks

In general, this type of resource allocation problem is NP-hard. In order to provide a comparison, we use binary integer

linear programming with linear constraints to find the optimal schedule. This formulation is then amenable to solution with a commercial binary solver that can deal with a linear constraint set. The notation we use for this is described in Table 5. The formulation we provide is extensible to more complicated scenarios, but here has been written for a scenario where each node can only process one task at a time, the compute time on each node is known deterministically, and all tasks are identical.

The formulation is shown in Equations 2 - 10. 2 states that the objective to be minimized is f, the final end time of all jobs. 3 requires that f be at least as large as f_t, the task finish time, for all tasks t. Because f is being minimized, this will ensure that it takes a value equal to the largest f_t. Equation 4 requires that each task $t \in \mathcal{T}$, is assigned to exactly one compute node. Equation 5 ensures that no task starts before it is available. Equation 6 states that for each task t, the duration $f_t - s_t$ will be the duration d_n for the node that the task is allocated to. Note that for every node the task is not allocated to, $x_{tn} = 0$ and so $x_{tn}d_n = 0$. Equations 7 and 8 ensure that tasks do not overlap if they are on the same node. 7 specifies that if tasks t and u are both performed on node n, then either y_{tun} or y_{utn} take the value 1, specifying that one occurs before the other. 8 requires that when $y_{tun} = 1$ for some node n, the start time of task u must be no earlier than the finish time of task t. When all of the relevant y_{tun} values are 0, the constraint becomes $s_u - f_t \geq -M$. M has been defined such that this constraint will be trivially satisfied for any values of s_u and f_t. The final constraints, 9 and 10, ensure that the binary variables will take appropriate values.

$$\min f \tag{2}$$

subject to

$$f_t \leq f \qquad \forall t \in \mathcal{T} \tag{3}$$

$$\sum_{n \in \mathcal{N}} x_{tn} = 1 \qquad \forall t \in \mathcal{T} \tag{4}$$

$$s_t \geq a_t \qquad \forall t \in \mathcal{T} \tag{5}$$

$$f_t = s_t + \sum_{n \in \mathcal{N}} x_{tn} d_n \qquad \forall t \in \mathcal{T} \tag{6}$$

$$y_{tun} + y_{utn} \geq x_{tn} + x_{un} - 1 \qquad \forall t, u \in \mathcal{T}, n \in \mathcal{N}, t < u \tag{7}$$

$$s_u - f_t \geq -M \left(1 - \sum_{n \in \mathcal{N}} y_{tun}\right) \qquad \forall t, u \in \mathcal{T}, n \in \mathcal{N} \tag{8}$$

$$x_{tn} \in \{0, 1\} \qquad \forall t \in \mathcal{T}, n \in \mathcal{N} \tag{9}$$

$$y_{tun} \in \{0, 1\} \qquad \forall t, u \in \mathcal{T}, n \in \mathcal{N} \tag{10}$$

3.3 Simulation results

Each workload is tested with three different resource allocation techniques and this is compared against the calculated Optimal solution. The heuristics are (1) random allocation of tasks to nodes, averaged over ten random runs, (2) Best Node, always assigning all jobs to the fastest node, (3) IoT, our workflow manager and (4,5) Bidding function based estimators and auctioneers as in Section 2.4. Table 6 shows the final completion time of the last task each method for each test case. Note that for instances 7 and 8, the MIP approach was unable to find the optimal solution within two hours on a laptop with a 2.60 GHz processor using IBM® ILOG® CPLEX® Studio. For these cases, we have reported the best solution found within two hours.

Table 5: Notation for optimisation model

Inputs	
$\mathcal{N}$	The set of all nodes available for computation.
$\mathcal{T}$	The set of all tasks requiring computation.
d_n	The duration required to compute a task on node n, $n \in \mathcal{N}$.
a_t	The time at which task $t, t \in \mathcal{T}$ becomes available for computation.
M	A large number, greater than the best final completion time.
Variables	
s_t	The start time of task t, $t \in \mathcal{T}$.
f_t	The finish time of task t, $t \in \mathcal{T}$.
f	The finish time of the entire set of tasks, $\max_{t \in \mathcal{T}} f_t$.
x_{tn}	A binary variable that takes the value 1 if task t is allocated to node n, 0 otherwise.
y_{tun}	A binary variable that takes the value 1 if tasks t and u are allocated to node n and task t starts before task u, 0 otherwise.

Instance 5 is the only workload where IoT does not come closest to the optimal solution. The IoT approach is not aware of the run times a priori and so, when all tasks are available at the same time, it will assign the first three tasks based on their calculated bid only. Thus in instance 5 – 8 each node will be allocated at least on task each, as its bid will become the largest once the other nodes begin the execution of tasks assigned to them. Every such workload of at least three tasks will therefore take at least 1.753 seconds (the run time on Node 3). For all other workload instances, the IoT approach comes closest to the optimal solution, followed by Best Node, with Random clearly the worst performer. This demonstrates the value of dynamically switching allocation between nodes rather than relying on static allocation to a node.

The first four instances, where a new task becomes available every $1/12$ of a second, are the more realistic cases, where tasks become available over time rather than all at once. For these cases, the IoT approach is always within 9% of the optimal solution, and the percentage gap decreases as the number of tasks increases. Bidding function based workflow manager results are shown in the 6^{th} and 7^{th}column of Table 6. We have chosen the resource providers to have some initial bidding functions, rather than starting from no function at all, without doing so they would underperform in these tests. With this pre-training their performance is similar to IoT or Best Node mechanisms. The IoT mechanism is faster than either bidding function methods. It can be clearly seen that the machine learning based bidding functions fin-

ish tasks faster than the simple bidding functions. These results suggest that in actual operation, where run times are not known in advance and many tasks are appearing over time, the workflow manager will outperform static allocation and may provide close to optimal allocation.[1]

Table 6: Last task execution time in seconds for all resource allocation mechanisms for each workflow.

Test	Optimal	Random	Best Node	IoT	$f_{bid,ML}$	f_{bid}
1	1.001	7.242	1.098	1.085	1.079	1.128
2	2.001	14.375	2.113	2.084	2.150	2.255
3	4.001	27.839	4.143	4.084	4.320	4.512
4	8.000	54.942	8.202	8.084	8.641	9.044
5	0.560	6.978	1.015	1.753	1.079	1.128
6	1.099	13.991	2.030	1.753	2.160	2.256
7	2.146[1]	28.630	4.059	3.720	4.320	4.512
8	5.007[1]	55.560	8.118	7.780	8.639	9.024

4 Conclusions and future work

. We present the IoT workflow manager and the MLAH approach, a system and framework for the allocation of tasks across IoT devices, Edge devices and Cloud computing resources with the ability to differentiate between otherwise homogeneous nodes based upon their internal state. Experiments and evaluations presented show that using the aforementioned hybrid machine learning, economic resource allocation approach we are able to allocate tasks to resources in a manner that results in outcome that are near the calculated optimal, which is a substantial improvement on random allocation, and in most cases an improvement over the other heuristics tested. Using simple machine learning models such as linear regression for allocation scheme presented in this work makes the latter much faster, scalable and lightweight making it perfectly aligned for IoT environments with small tasks/workloads.

We are continuing the development of the IoT workflow manager with a focus on security, scalability and real-time deployment. We are also experimenting with improving the scalability of our resource allocator through a publish subscribe model for execution history and bid data. The work so far assumes a trustworthy environment and that all nodes are truthful in their responses. We acknowledge that more work is needed to identify when nodes are not truthful in their responses or are indeed malicious in their submission of tasks. This is a point of concern that must be addressed in future research.

References

[1] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 945–953.

[2] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 49–62. [Online]. Available: http://doi.acm.org/10.1145/1814433.1814441

[3] F. Jalali, O. J. Smith, T. Lynar, and F. Suits, "Cognitive iot gateways: Automatic task sharing and switching between cloud and edge/fog computing," in *Proceedings of the SIGCOMM Posters and Demos*, ser. SIGCOMM Posters and Demos '17. ACM, 2017, pp. 121–123.

[4] P. G. Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, Sep. 2015.

[5] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014. [Online]. Available: 10.1145/2677046.2677052

[6] F. Xia, F. Ding, J. Li, X. Kong, L. T. Yang, and J. Ma, "Phone2cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing," *Information Systems Frontiers*, vol. 16, no. 1, pp. 95–111, Mar 2014. [Online]. Available: https://doi.org/10.1007/s10796-013-9458-1

[7] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proceedings of the Sixth Conference on Computer Systems*, ser. EuroSys. New York, NY, USA: ACM, 2011, pp. 301–314. [Online]. Available: http://doi.acm.org/10.1145/1966445.1966473

[8] H. Qian and D. Andresen, "Jade: Reducing energy consumption of android app," *International Journal of Networked and Distributed Computing*, vol. 3, no. 3, pp. 150 – 158, August 2015.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[10] T. M. Lynar, Simon, R. D. Herbert, and W. J. Chivers, "Reducing energy consumption in distributed computing through economic resource allocation," *the International Journal of Grid and Utility Computing (IJGUC)*, vol. 4, no. 4, pp. 231–241, 2013.

[11] A. Kopytov, "Sysbench v1.09," https://github.com/akopytov/sysbench/, 2017.

[1] refers to cases where the optimal solution is not found within the prescribed time-limit.