IJICTR International Journal of Information & Communication Technology Research

Volume 8- Number 1- Winter 2016 (43-49)

A Priority-based Fast Optimal Computation Offloading Planner for Mobile Cloud Computing

Mohammad Goudarzi
Faculty of Computer Engineering
Iran University of Science and Technology (IUST)
Tehran, Iran
mohammad_goudarzi@alumni.iust.ac.ir

Dr. Zeinab Movahedi
Faculty of Computer Engineering
Iran University of Science and Technology (IUST)
Tehran, Iran
zmovahedi@iust.ac.ir

Prof. Guy Pujolle
Laboratoire d'Informatique de Paris 6 (LIP6)
University of Pierre & Marie Curie (UPMC)
Paris, France
guy.pujolle@lip6.fr

Received: November 11, 2015- Accepted: March 7, 2016

***Abstract*— Today's advances of mobile technologies in both hardware and software have pushed the vast utilization of mobile devices for diverse purposes. Along with this progress, today's mobile devices are expected to perform various types of applications. However, the energy challenge of mobile devices along with their limited computation power act as a barrier. To address this deficiency, mobile cloud computing has been proposed in which cloud resources are used to extend mobile devices' capabilities. However, due to varying conditions of wireless channel in terms of connectivity and bandwidth, an online offloading mechanism is required which may lead to high decision time and energy. To address this challenge, we propose a priority-based fast computation offloading mechanism which finds the optimal offloading solution based on a modified branch-and-bound algorithm. Results of intensive simulation and testbed experiments demonstrated that our proposal can outperform all existing optimal counterparts in terms of energy consumption and execution time.**

Keywords- Mobile Cloud Computing, Computation Offloading, Optimal Partitioning

I. INTRODUCTION

Due to nowadays advancements in mobile applications and technologies, mobile devices have become as one inseparable part of our today's life. This development has resulted in using mobile devices for more computation intensive programs, which may not be efficiently executable by a mobile device. Moreover, the execution of such applications may exhaust the battery lifetime of energy-constrained mobile devices [1, 2]. The mobile cloud computing, which consists in employing the cloud capabilities by a mobile device [3], can address these performance and energy consumption concerns. In this regard, some parts of computation intensive applications are

transferred from the mobile device to the cloud, which subsequently returns the processed information as a result. This process is referred to as mobile cloud computation offloading (MCCO) [4-7].

Due to the variable nature of network bandwidth and variety of applications' characteristics in terms of quality of service requirements and data amount required to be transferred for remote execution, offloading of some parts of an application may be less or more beneficial in different times. Consequently, recent works concentrate on developing decision planners in order to specify appropriate constituent parts of application (called application's units) for offloading. To achieve this, application's units are primarily extracted using static or dynamic analysis of the application code or in a predefined manner based on object, method, thread, component or service granularity. Then, the weighted relation graph (WRG) of application is constructed to describe constituent application's units and their respective invocations. The graph is profiled in an online or offline manner based on the cost of local and remote execution of each vertex and their corresponding invocation cost. Finally, the offloading problem is resolved by alternating between local and remote execution of different application's unit in order to discover a combination which minimizes the time and energy of offloading.

Due to the importance of offloading decision, a number of works concentrated on proposing adequate planners to resolve offloading partitioning problem. These works can be classified in optimal and near-optimal planners according to the primary objective of their decision planner. The main concern of optimal offloading decision planners [8-12] is to find the optimal solution regardless of time devoted to make such decision. However, near-optimal proposals [13-16] attempt to find a partitioning as close as possible to the optimal solution in order to reduce the decision time. However, they may sometimes lead to an inappropriate outcome far from the optimal result since these algorithms suffer from their non-deterministic nature and local optimum problem. Consequently, we believe that optimal solutions should be preferred as long as their decision time remains reasonable.

In this paper, we propose a priority-based fast optimal computation offloading planner, called PFO, based on branch-and-bound algorithm. The main goal of PFO is to find the optimal solution for diverse application size in a timely manner.

The rest of paper is organized as follows. Section 2 provides an overview of the current literature in mobile cloud offloading. Section 3 presents our proposed PFO solution. Section 4 describes the evaluation environment and reports the obtained results. Finally, section 5 concludes the paper and draws future works.

II. RELATED WOKS

Integer Linear Programming (ILP) is employed as optimal offloading decision planner in a number of existing proposals [8-12]. Mainly, **MAUI** [8] uses the ILP to optimize the energy consumption and execution time of application. The weighted relation graph of MAUI is composed of application's methods extracted in a pre-defined manner and weighted based on energy, CPU cycle and bandwidth. **CloneCloud** [9] employs the ILP to minimize the application's execution energy based on a thread-based application relation graph extracted through static analysis. The ultimate goal of **Mobile Augmentation Cloud Services** (MACS) [10] is to minimize the energy consumption, memory usage and execution time of application. The relation graph is constructed through static analysis of application's services profiled based on service type (i.e. offloadable or not), code size and memory cost, just to mention a few.

In order to reduce the decision time of ILP-based offloading solutions, **Branch and Bound Application Partitioning** (BBAP) [11] attempts to find the optimal solution using the branch-and-bound algorithm. BBAP creates the weighted object relation graph of application through static analysis and online profiling. It employs the depth-first-search strategy to explore the search space. The bounding value of BBAP proposal is initialized based on the cost of graph min-cut in the least possible bandwidth, obtained by Stoer-Wagner algorithm [13].

Since the existing optimal planners are not capable to solve the offloading problem in a timely manner especially for higher scale graphs, some recent proposals have been concentrated on near-optimal solutions [14-17].

Genetic Algorithm based computation offloading algorithm (GACO) [16] applies the genetic algorithm on a weighted relation graph composed of services workflow obtained by static analysis and online profiling. GACO uses random initial population and a fitness function minimizing the energy consumption and execution time. GACO crossovers the genes of two parents by correspondingly comparing them regarding their sensibility to mobility and fault occurrence. It mutes a gene with a probability proportional to its fault occurrence and mobility sensibility. These steps are repeated for a pre-defined iteration runs.

The objective of **Dynamic offloading algorithm** (DOA) [18] is to find a tradeoff between energy consumption and execution time using 1-opt local search. DOA uses static analysis and bandwidth-based online profiling. The employed 1-opt local search strategy is initialized with a feasible solution from which other solutions are derived by changing the state of one node and calculating its cost. Ultimately, the solution with minimal cost is returned when the stop condition is reached.

Min-Cut Greedy Application Partitioning (MCGAP) [11] performs on object granularity and extracts the min-cut of graph through static Stoer-Wagner algorithm for both least possible and current bandwidth. Nodes assigned to remote server in the least possible bandwidth and to local site in the current bandwidth are planned for execution in their corresponding selected site.

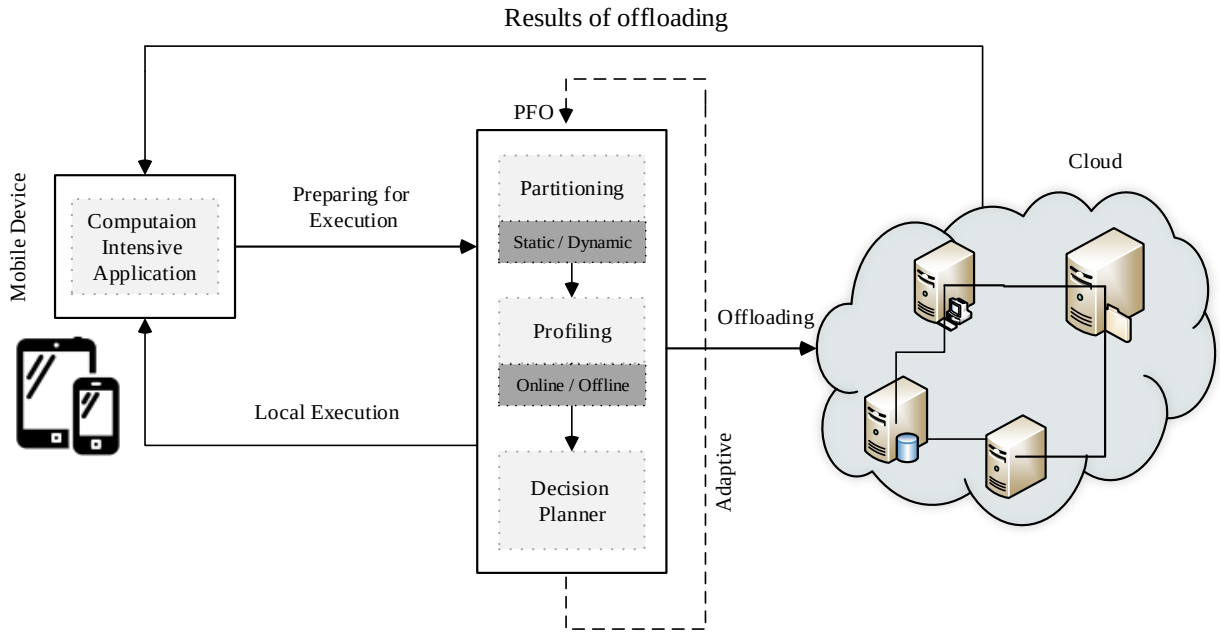

Fig.1. The proposed PFO model of mobile cloud computing

Remained nodes are assigned to the less-cost execution site according to decreasing order of their weights.

Although the aforementioned near-optimal proposals can decrease the offloading decision time for large scale graphs, the calculated outcome may differ significantly from the optimal solution. In this paper, we aim at proposing a fast offloading approach capable to achieve the optimal solution in a same order of decision time as near-optimal solutions.

III. PRIORITY-BASED FAST OPTIMAL COMPUTATION OFFLOADING PLANNER

In this section, we propose our priority-based computation offloading solution, called PFO, which is capable to obtain the optimal offloading solution in a timely manner. We first model the offloading issue as an optimization problem to minimize the energy consumption and execution time of application. Then, we describe our proposed decision planner for solving the computation offloading model. Fig. 1 illustrates an overview of our proposed PFO.

A. Offloading Model

In our proposed PFO, the partitioning module extracts the application's units through static analysis of component level granularity. The application's units and their respective relations are represented as a relation graph in which vertices and edges represent constituent components and their invocations, respectively. The profiling module constructs the WRG of application in online manner, considering the available bandwidth, CPU processing speed, component's instructions number, transmission power and CPU power. The execution time and energy consumption are modeled as follows.

1) Execution time model

Application's execution time is obtained through execution time of each component with respect to its execution site (Local or Remote) in addition with the invocation time. The execution time of application is formulated as follows:

$$T = \sum_{i=1}^{n} \left((1 - x_i) \times t_i^{loc} + x_i \times t_i^{rem} \right) + \sum_{i,j=1}^{n} \left(|x_i - x_j| \right) \times t_{i \to j}^{rem} \quad (1)$$

Where x_i is the exclusion factor, representing the local (x_i = 0) or remote (x_i = 1) execution site of each component. Local execution time (t_i^{loc}) of each component is defined based on number of instructions within each component (ins_num_i) divided by processing speed of CPU ($proc\_speed^{CPU}$). Furthermore, the remote execution time (t_i^{rem}) of each component is calculated by dividing the local execution time of each component by cloud speed up factor (k). The speed up factor represents the ratio of remote to local site processing speed. Consequently, the component execution time in local and remote site can be formulated as follows:

$$t_i^{loc} = \frac{ins\_num_i}{proc\_speed^{CPU}} \quad (2)$$

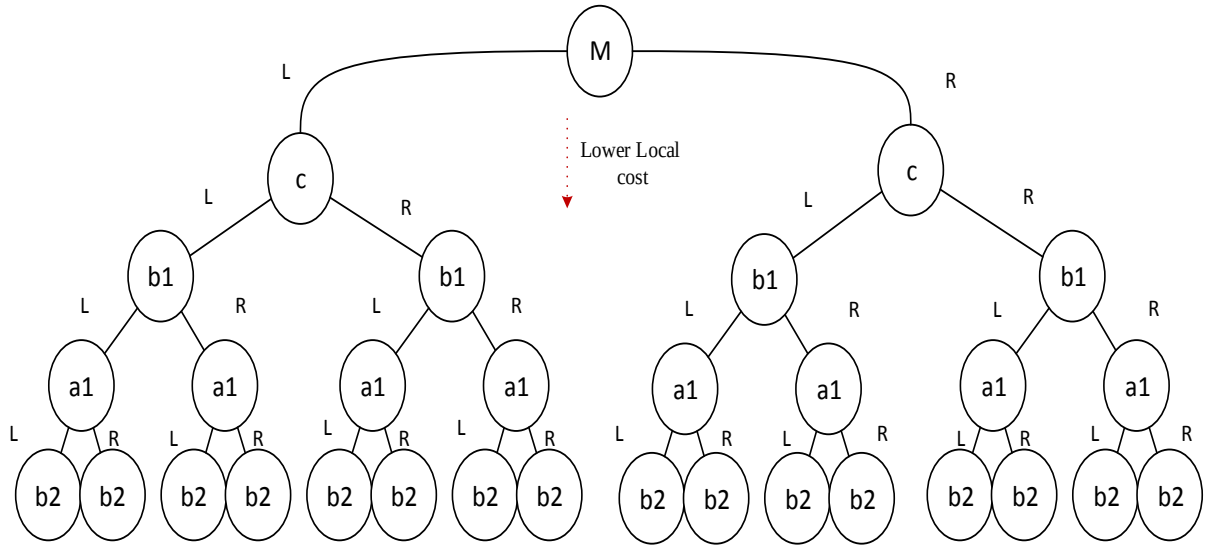

Fig 2. An example of PFO's branchin rule

$$t_i^{rem} = \frac{t_i^{loc}}{k} \qquad (3)$$

The invocation time between two linked components ($t_{i\to j}^{inv}$) is obtained by dividing the transmitted data between two components to the available transmission bandwidth. It is worth mentioning that the invocation time between two components in the same site is assumed to be negligible and equal to zero. The invocation time is defined in the following:

$$t_{i\to j}^{inv} = \frac{datasize_{i\to j}}{bw^{transmit}} \qquad (4)$$

2) Energy consumption model

The offloading energy consumption is derived from the energy consumed for executing components selected to be run in the mobile device and the energy devoted for invocations between related components executed in different sites. The energy consumption of processing components in the cloud side is considered equal to zero from the view of the mobile device. Hence, the offloading energy consumption is formulated as follows:

$$E = \sum_{i=1}^{n}\left((1 - x_i) \times e_i^{loc}\right) + \sum_{i,j=1}^{n}\left(|x_i - x_j|\right) \times e_{i\to j}^{inv} \qquad (5)$$

The local energy (e_i^{loc}) and invocation energy ($e_{i\to j}^{inv}$) are calculated as follows:

$$e_i^{loc} = t_i^{loc} \times p^{CPU} \qquad (6)$$

$$e_{i\to j}^{inv} = t_{i\to j}^{inv} \times p^{transfer} \qquad (7)$$

B. Proposed Branch-and-Bound Decision Planner

The proposed decision planner in PFO employs an optimized branch-and-bound which is capable to find the optimal offloading solution in a timely manner. In the following, we present our proposed strategy for branching rule, bounding function and search steps of PFO.

In the branching rule step, the WRG is transformed to a tree in which each branch represents a possible combination of execution sites for different application's components. To achieve this, the tree is initialized from an empty root. Then, two copies of each component within the WRG are added to each leaf in a tree level, one for remote (x_i = 1) and one for local (x_i = 0) execution of that component. In order to reduce the time of tree exploration in the search step, nodes of WRG are added to the tree in descending order of their weights, placing nodes with higher weights closer to the tree root. This latter allows our proposed PFO to prune inappropriate branches as soon as possible in the search step, which can result in reducing the time of offloading decision. Fig. 2 depicts a tree constructed by the proposed branching rule. In this example, it is assumed that WRG has four components, including c (the most computation intensive), a1, b1 and b2 (the least one).

Since the goal of offloading is to find a solution with less cost compared to the local execution, we initialized our bounding value with the cost of local execution. The time complexity for updating the bounding value is equal to Θ(1) which allows to further reduce the decision time.

To traverse the formed search tree, we employed the Depth First Search (DFS) strategy during which the cost of each branch is calculated according to the abovementioned offloading cost model. During the

TABLE I. Graph specifications

Random Graph	Nodes	Edges
R1	5	7
R2	10	30
R3	15	60
R4	25	230

TABLE II. Simulation parameters

Variables	values
p^{CPU} (W)	0.4
$p^{Transfer}$ (W)	0.7
K	3
$bw^{transmit}$ (KB/s)	10 - 100

exploration, a branch may be pruned if its cost becomes higher than the bounding value. A branch is considered as a possible solution if it is explored entirely without being pruned. In such a case, the bounding value is updated by the cost of this branch, which is evidently less than the previous bounding value. As the result of the bounding value update, the search strategy becomes able to prune other inadequate branches in the faster time and hence to reduce the decision time. This process is executed until the search space is traversed completely.

IV. EVALUATION

We evaluated the performance of our proposed PFO using both simulations and real testbed. The main objective of simulation experiments was to compare the efficiency of PFO with other optimal decision planners existing in the literature under different input parameters. Our real testbed analysis aimed at studying the applicability of our proposed solution considering the limitation and characteristics of real world in terms of application specifications, mobile device properties, network connection quality and bandwidth. In the following, we describe each evaluation study and the obtained outcome.

A. Simulation Results

With regard to the simulation study, we compared the effectiveness of the proposed PFO with ILP and BBAP counterparts using MATLAB version R2013 on a machine with 2.2 GHz Intel core i7 CPU and 8 GB of RAM. As our simulation test cases, we generated a set of random graphs as specified in table I, representing various relation graphs to fit with different possible applications. Generated relation graphs are weighted based on uniform distribution. Moreover, for each random graph, simulations are run 33 times and their average is represented as the result with a 95% confidence interval. Simulation parameters are summarized in table II.

For this evaluation sets, we studied the impact of application graph scale and transmission bandwidth on the execution time and energy consumption of application. These analysis are described hereafter.

TABLE III. Decision time (S)

Random Graph	PFO	BBAP	ILP
R1	0.019	0.019	0.021
R2	0.028	0.0367	0.085
R3	0.039	0.1960	28.91
R4	0.153	13	-

TABLE IV. Decision energy (J)

Random Graph	PFO	BBAP	ILP
R1	0.007	0.007	0.008
R2	0.009	0.014	0.034
R3	0.013	0.078	28.91
R4	0.049	5.2	-

1) Scale of relation graph impact

In this experiment, the transmission bandwidth is fixed to 100 Kbytes/s. The decision time and decision energy of PFO are compared to its counterparts in table III and table IV, respectively. As we expect, the decision time and decision energy increase as the size of graph increase. The numerical results demonstrate that PFO outperforms both BBAP and ILP in decision time and decision energy. Also, as the relation graphs become larger, the supremacy of PFO to its counterparts are more highlighted. This latter is achieved by means of PFO's intelligent branching rule and bounding function.

2) Transmission bandwidth impact

In this experiment, the efficiency of PFO and BBAP are compared in different transmission bandwidths, ranging from 10 Kbytes/s to 100 Kbytes/s. We evaluate the performance of R3 and R4 in terms of execution time and energy consumption. According to Fig. 3 to Fig. 6, PFO outperforms BBAP for both R3 and R4, independently of underlying bandwidth.

As the bandwidth increases, both energy consumption and execution time follow a descending pattern, However, PFO shows better offloading gain. This latter is due to better pruning of the search space, which results in reduction of execution time and energy consumption.

B. Testbed Results

For our real use case experiment, we implemented the proposed PFO decision planner on a Galaxy S6 edge as the mobile device. In addition, we employed Zenbook with 2.9 GHz Intel Core i7 CPU and 8 GB of RAM as the remote server. As a sample mobile application, we implemented Fibonacci algorithm on Android Platform since it can lead to high execution time and energy for high values of input parameter *n*. For our analysis, we run the Fibonacci algorithm using both local execution strategy and PFO solution for different number of input values n, ranging from 35 to 51. Table V and VI respectively demonstrate the

TABLE V. Execution time results for fibonacci testbed (S)

Method	*n = 35*	*n = 38*	*n = 42*	*n = 45*	*n = 48*	*n = 50*	*n = 51*
Local	0.3	0.98	5.68	23.40	119.82	-	-
FACO	0.5	0.67	2.33	9.04	43.92	120.10	183.11

TABLE VI. Energy consumption results for fibonacci testbed (J)

Method	*n = 35*	*n = 38*	*n = 42*	*n = 45*	*n = 48*	*n = 50*	*n = 51*
Local	0.12	0.4	2.27	9.36	47.92	-	-
FACO	0.15	0.15	0.15	0.15	0.15	0.15	0.15

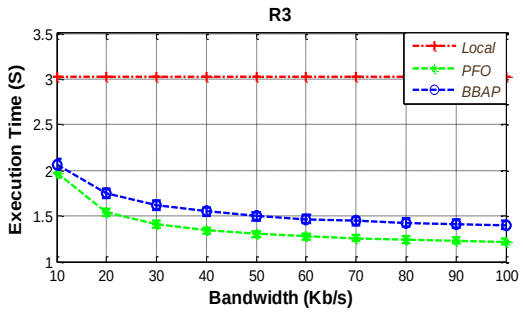

Fig 3. Bandwidth Impact on execution time using R3

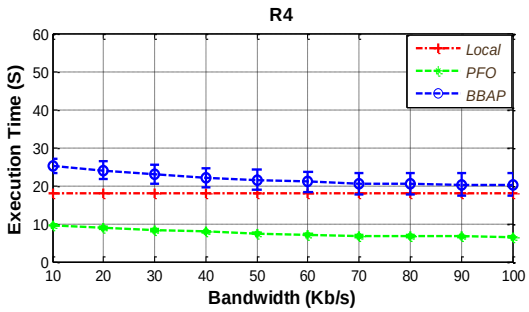

Fig 5. Bandwidth Impact on execution time using R4

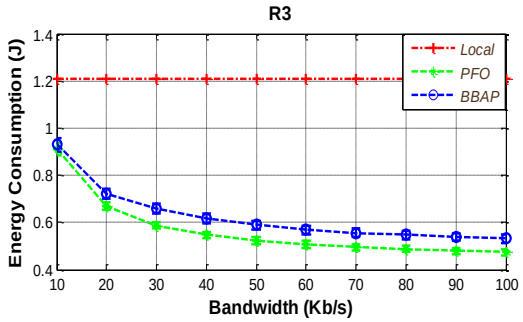

Fig 4. Bandwidth Impact on energy consumption using R3

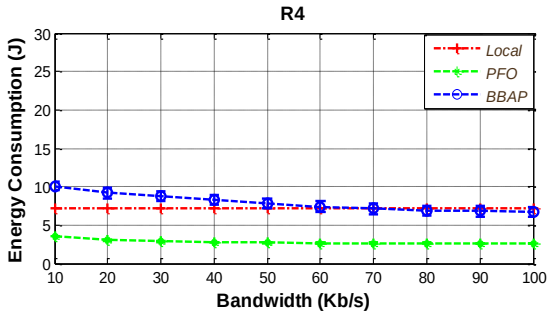

Fig 6. Bandwidth Impact on energy consumption using R4

execution time and energy consumption obtained by PFO and local execution.

Table V illustrates that as the value of *n* increases, execution time of both PFO and local execution increase, however PFO's execution time outperforms the local execution time. Moreover, the mobile device cannot run Fibonacci for values of *n* larger than 48 due to RAM constraints.

Table VI shows that the local energy consumption has the same trend similar to local execution time, with an increasing trend for higher values of *n*. However, PFO's energy consumption remains steady with different values of *n*. This latter is due to the consideration of the energy consumption only from the mobile device perspective. Therefore, the energy consumption of an application execution remains intact from when all applications units are decided to be executed in the remote server.

V. CONCLUSION

In this paper, we proposed a priority-based computation offloading solution for mobile cloud computing which is capable to obtain optimal application partitioning in a timely manner. The proposed solution uses the branch-and-bound algorithm as its decision planner. It proposes an appropriate branching rule and bounding function in order to reduce the search space and optimize the resulted decision time and energy. The effectiveness of the proposed approach is evaluated using both simulations study and real tested experiment. Results demonstrated that the proposed solution outperforms other existing optimal counterparts in terms of both execution time and energy consumption.

As future work, we expect to study the impact of intermittent wireless connectivity and significant bandwidth changes occurred during initialization of offloading process and execution of application.

Moreover, we intent to propose a mobility-aware offloading solution to optimize the effective outcome of offloading decision.

ACKNOWLEDGMENT

This work has been supported by the Center for International Scientific Studies and Collaboration (CISSC) and French Embassy in Iran.

REFERENCES

[1] Z. Sanaei, S. Abolfazli, A. Gani, and R. H. Khokhar, "Tripod of requirements in horizontal heterogeneous mobile cloud computing," in Proc. 1st International Conference onComputing, Information Systems, and Communications (CISCO'12), Singapore, May 2012.

[2] B. Rajesh Krishna, "Powerful change part 2: reducing the power demands of mobile devices,"IEEE Pervasive Computing, vol. 3, no. 2, pp. 71–73, 2004.

[3] H. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches,"Wireless Communications and Mobile Computing, 2011.

[4] Z. Sanaei, S. Abolfazli, A. Gani, and M. Shiraz, "SAMI: Servicebased arbitrated multi-tier infrastructure for mobile cloud computing," in Proc. IEEE 1st International Conference on Communications in China Workshops (ICCC), Beijing, China, Aug. 2012, pp. 14–19.

[5] H, Flores, and S. N. Srirama. "Mobile cloud middleware."Journal of Systems and Software 92, 2014, pp. 82-94.

[6] 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", IEEE proceedings on INFOCOM, 2012, pp. 945-953.

[7] M. Shiraz, A. Gani, A. Shamim, S. Khan, and R. W. Ahmad. "Energy Efficient Computational Offloading Framework for Mobile Cloud Computing." Journal of Grid Computing 13, no. 1, 2015, pp 1-18.

[8] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu and R. Chandra, "MAUI: making smartphones last longer with code offload", in Proceedings of the 8th international conference on Mobile systems, applications, and services, 2010, pp. 49-62.

[9] 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, 2011, pp. 301-314.

[10] D. Kovachev, Y. Tian, and R. Klamma, "Adaptive Computation Offloading from Mobile Devices into the Cloud", IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2012, pp. 784-791.

[11] J. Niu, W. Song, and M. Atiquzzaman, "Bandwidth adaptive partitioning for distributed execution optimization of mobile applications", Journal of Network and Computer Applications, vol. 37, 2013, pp. 334–347.

[12] D. Yao, C. Yu, H. Jin, and J. Zhou, "Energy Efficient Task Scheduling in Mobile Cloud Computing", in Network and Parallel Computing, vol. 8147, 2013, pp. 344-355.

[13] M. Stoer and F. Wagner. "A simple min-cut algorithm." Journal of the ACM (JACM) 44.4, 1997, pp. 585-591.

[14] K. Sinha and M. Kulkarni, "Techniques for fine-grained, multi-site computation offloading, in Cluster", 11th IEEE/ACM International Symposium on Cloud and Grid Computing (CCGrid), 2011, pp. 184-194.

[15] H. Wu, D. Huang, "Modeling multi-factor multi-site risk-based offloading for mobile cloud computing.", In 10th international conference on network and service management (CNSM). IEEE, 2014, pp. 230–235

[16] S. Deng, L. Huang, J. Taheri, A. Zomaya, "Computation Offloading for Service Workflow in Mobile Cloud Computing.", IEEE Transactions on Parallel and Distributed Systems, 2014, DOI:10.1109/TPDS.2014.2381640.

[17] N. Kaushi and J. Kumar, "A computation offloading framework to optimize energy utilisation in mobile cloud computing environment," Int. Journal of Computer Applications and Information Technology, vol. 5, no. 2, 2015 pp. 61-69.

[18] D. Huang, P. Wang, and D. Niyato. "A dynamic offloading algorithm for mobile computing." Wireless Communications, IEEE Transactionsm, 2012, pp. 1991-1995.

Mohammad Goudarzi received the M.Sc degree in information technology (Network scinece) from Iran University of Science and Technology (IUST), Tehran, Iran, in 2015. Currently, he is a research assisstant at PostIP Labatorary in IUST. His research interests include mobile cloud computing, optimization problems, wireless ad hoc networks, wireless sensor networks, and green computing.

Zeinab Movahedi received the M.Sc and Ph.D degrees in computer networks and telecommunications from the University of Pierre and Marie Curie (Paris 6), Laboratoire d'Informatique de Paris 6 (LIP6), Paris, France, in 2011. She is currently an Assistant Professor at IUST, and a Member of the PHARE Research Team of LIP6, UPMC – Paris 6, France. Her research interests include autonomic networking, policy-based network management, mobility and resource management in wireless networks, network security, performance evaluation, and quality-of-service support.

Guy Pujolle is currently a Full Professor at the LIP6, University of Pierre and Marie Curie, Paris, France, and a member of The Royal Physiographical Academy of Lund, Sweden. He is the French representative at the Technical Committee on Networking at IFIP. Guy Pujolle is a pioneer in high-speed networking having led the development of the first Gbit/s network to be tested in 1980. He was participating in several important patents like DPI (Deep Packet Inspection), virtual networks or virtual access points. He is the cofounder of QoSMOS (www.qosmos.fr), Ucopia Communications (www.ucopia.com), EtherTrust (www.ethertrust.com), Virtuor (www.VirtuOR.fr), and Green Communications (www.greencommunications.fr).

This Page intentionally left blank.