Advances in Aeronautical Science and Engineering
ISSN: 1674-8190

Energy-Efficient Mobile Application Execution Through a Mutual-Benefit Approach

Sophie Collins[1], Daniel Evans[2], Claire Fisher[3], and Mark Grant*[4]

[1] Department of Neuroscience, University of Zurich, Switzerland
[2] Institute of Cognitive Science, University of Edinburgh, UK
[3] School of Psychology, University of Cape Town, South Africa
*[4] Department of Neuroscience, University of Zurich, Switzerland

ABSTRACT

The MUTUAL-BENEFIT approach exploits optimal task offloading, task scheduling, resource allocation, and provider selection process to execute the mobile cloud applications. The proposed approach enabled mobile cloud environment ensures the seamless application execution resulting in extending the battery lifetime and the optimal profit. The mobile cloud task scheduling and resource allocation process schedules the offloaded tasks and allocates the resources merely based on the availability and the resource requirements. The additional consideration of the proposed algorithm in mobile cloud environment facilitates both the mobile users and the providers in reducing the burden of application execution and mitigating the processing complexity respectively. The MUTUAL -BENEFIT creates the greater impact on tackling the battery constraint and manipulating dynamic numerous user requests with high profit. The proposed algorithm retains the energy level in mobile devices by 10%, minimizes the response time by 12% and application completion time by 20%, and maximizes the profit of the cloud service provider by 11% for mobile applications.

KEYWORDS: MUTUAL-BENEFIT approach, Mobile applications, Algorithm, Energy, Environment.

1. INTRODUCTION

SLA is a crucial consideration of both the perspectives of the mobile end user and the cloud provider. Most of the conventional methods discuss the task scheduling, resource allocation and load balancing on MCC either on the end user or cloud service provider. In MCC environment, millions of mobile users submit the same application request at the same time. Therefore, optimal scheduling and allocation are critical to make the significant impact on both the end user side and the provider side. The cloud service provider makes SLA with the end user's requirements, where the specific, measurable characteristics of SLA are end user's mobile device energy and response time. With the aim of satisfying SLA of end user convenience in terms of long battery life time, quick response and simultaneously maximizes the profit of the service provider. The proposed approach enhances ACO algorithm and optimizes task offloading, task scheduling, resource allocation, and provider selection in an MCC environment to satisfy SLA of the end user and to enhance the profit of the provider. SLA based optimization selects optimal cloud resources for compute-intensive mobile application execution. In addition, this chapter presents the implementation results of the MUTUAL BENEFIT approach with baseline NTGO, and E-LHEFT approaches.

2. AN OVERVIEW OF EXISTING OPTIMIZATION TECHNIQUES

ACO meta-heuristics [1] dynamically schedules the workload based on the current workload and resource availability. ACO based load balancing [2] considers the routing packets as the ants in the cloud environment. It replaces the routing tables with a probability value of pheromone tables which contains the information of pheromone value and incremental pheromone update. EAPA approach [3] addresses the minimal delay problem by applying an initial task scheduling algorithm. It migrates the tasks for minimizing the device's energy using the rescheduling algorithm in a mobile cloud environment. Hence, the application migration degrades the performance of the system in the mobile cloud. NTGO framework [4] minimizes the device energy and improves the performance of response time by effective offloading decision, and increases provider's profit. DPOA [5] takes an offloading decision based on the optimal partitioning of an application. Even though conventional methods focus on the energy-based optimization in MCC environment, the optimization model is necessary to maintain the trade-off between performance and cost. Hence, the proposed approach contemplates the SLA objectives and profit of the provider as the major constraints. Also, the optimization method needs to achieve the QoS without SLA violations.

3. MUTUAL-BENEFIT SYSTEM MODEL

This section presents a MUTUAL-BENEFIT system model for providing cloud services for the consideration of optimal scheduling and allocation. It is assumed that the mobile devices comprise of poor processing capability, if it outsources the resource-hungry applications to the cloud. MCC environment consists of a set of similar applications (A) from various mobile users i = {1, 2,...m}, and cloud resources j = {1,2,...., n}. An

ISSN: 1674-8190

appropriate assignment of j ∈ cloud resources to $A_i \in A$ provides the optimal service to the end-user 'i'. In cloud server, scheduling manager segregates the applications into tasks (T_i). To select the optimal VM for T_i, it is essential to consider the task completion time (T_{ij}), load balancing (n_{ij} ($S\omega'(t)$)), and profit ($S\omega(t)$) in which n_{ij} represents the optimal load balancing factor.

Infrastructure Service Provider: Infrastructure Service Provider (ISP) is known as the virtual resource provider. ISP provides the virtual resources in terms of VMs to the Cloud Service Provider (CSP). CSP rents the VMs to end-users based on the amount charged by the ISP. Each VM resource has unique configurations of CPU, price, and memory.

Cloud Service Provider: CSP is also known as the service provider. CSP provides the rented virtual resources to the end-users for processing mobile applications in the cloud. It selects the best $S_p \in$ set of ISPs, and it furnishes the resources of S_p with execution services to improve user satisfaction level and its profit.

End-user: End-user must pay the amount to a service provider that depends on the SLA and received service utilization. The payment of the end -user is the revenue of CSP. SLA violation reduces the revenue of Ai, if the application takes longer time than average execution time. Thus, it is essential to consider both the energy cost and the revenue for maximizing the profit of the provider and satisfying the SLA objectives.

4. MUTUAL-BENEFIT METHODOLOGY

For instance, the Sudoku solver application contains a different number of cells based on the level of the application. The mobile device partially fills the cells in Sudoku solver application due to the energy constraint of the mobile device. The ThinkAir architecture based offloading manager monitors the energy model of the device to offload the resource intensive tasks in partially filled cells of the Sudoku solver application to the cloud server. In Sudoku solver application, empty cells are considered as the cloud tasks. Non-recursive dynamic programming based ACO method schedules the cloud tasks by selecting the SLA objectives based optimal VM resources. This method follows the basic function of ACO approach while identifying the best solution for task scheduling. Finally, the Bellman's theory-based utility function optimally allocates the resources to determine the solution for empty cells. The selected optimal VM resources enable the corresponding task to execute the solution to find the corresponding unfilled cells in Sudoku solver application. This approach is targeted to achieve load balancing of an application that also provides the long-lasting device battery. This optimal execution of MUTUAL-BENEFIT balances the objectives of both the end-user and the service provider. The proposed methodology of MUTUAL-BENEFIT in MCC is shown in Figure 1.

Optimal task offloading using ThinkAir architecture

The computation offloading aims to migrate the resource-intensive computations from a mobile device to the resource-rich cloud. It enhances the performance of mobile applications that are unable to execute in smartphones due to insufficient battery energy resources. The MUTUAL- BENEFIT approach employs the ThinkAir architecture [6] [7] [8] to make the offloading decision on the mobile cloud environment dynamically. Figure 2 shows the architecture of ThinkAir framework. Also, ThinkAir architecture supports to execute the dynamic programming in MUTUAL -BENEFIT, where the decision about recursive tasks is taken using the stored offloading information without re-execution. Further, it reduces the complexity of assigning tasks and finding optimal resources in MUTUAL-BENEFIT.

Dynamic programming based offloading method (DPOM)

The MUTUAL-BENEFIT exploits the ThinkAir architecture to divide the application into mobile and cloud tasks according to the mobile device energy. The execution controller of ThinkAir architecture implements Dynamic Programming based Offloading Method (DPOM) to quickly find the optimal partitioning (mobile and cloud tasks) between executing subcomponents of a mobile application for the mobile devices and the cloud server, taking account the CPU speed of the mobile device, network performance, mobile device energy, the characteristics of the application program and the efficiency of the cloud server.

The offloading decision is based on the dynamic programming method which exploits the decision information from previous offloaded tasks. By utilizing the dynamic programming method, the proposed approach explores the conditions of the device energy, and task complexity from the previously stored data along with the current status of the device, which facilitates the offloading process within an acceptable offloading time between the mobile device and the cloud server. DPOM solves the offloading optimization problem with much lower complexity ($O(n^2)$) than the Branch & Bound method ($O(2^n)$), while significantly reducing the execution time of mobile applications.

ISSN: 1674-8190

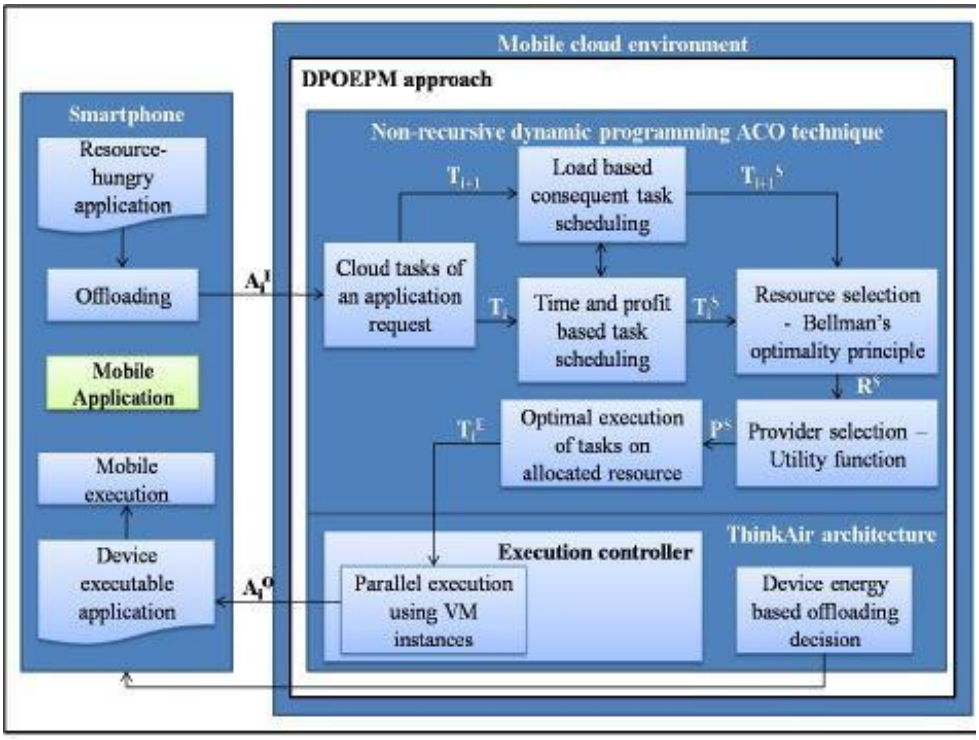

Figure 1: MUTUAL-BENEFIT approach in MCC

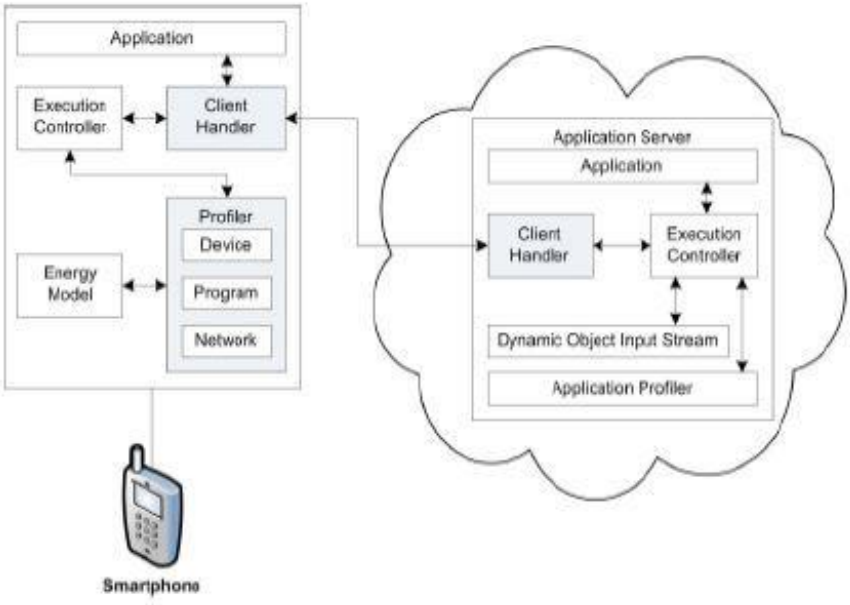

Figure 2: Architecture of ThinkAir framework

Satisfying SLA objectives via optimal task scheduling in the mobile cloud

The 'execution controller' of ThinkAir architecture executes the MUTUAL-BENEFIT algorithm in a remote server. The cloud service provider provides processing, memory, and communication resources to the mobile users based on the SLA. The main goal of the cloud provider is to satisfy the user convenience in terms of battery energy and response time while providing the service. The MUTUAL-BENEFIT considers SLA as an important factor while performing task scheduling and resource allocation. It employs the ACO technique to schedule the tasks optimally and executes the non-recursive dynamic programming with the support of ThinkAir architecture. Before selecting the corresponding optimal cloud resources to the task using ACO technique, the MUTUAL -BENEFIT obtains the cloud tasks of an application from the offloading manager.

ACO technique

The MUTUAL-BENEFIT approach follows the basic function of ACO technique while identifying the best solution for task scheduling. The non-recursive dynamic programming based ACO technique considers together of execution time, load balancing, and profit as Pheromone value to achieve SLA objectives. To reduce the

ISSN: 1674-8190

computation complexity, the MUTUAL-BENEFIT modifies the Brute-force search based ACO algorithm into Dynamic programming based ACO algorithm. Brute-force search based ACO technique degrades the QoS due to the optimal solution searching process based on $O(2^n)$ complexity of all combinations. Hence, the proposed approach exploits ACO with the dynamic programming of O(n) complexity. An optimal selection of each task' pheromone value based on the satisfaction of SLA objectives. The pheromone value updating depends on the optimal solution of an ant while mapping task into resource at a time (t). χ is the random variable that denotes the decaying parameter. The pheromone value of i^{th} task at updated time (T') can be formulated in equation 1.

$$\Upsilon_i T' = (1-\chi)\ \Upsilon_i(t) + \Delta\Upsilon_i(t,T') \quad \text{, where } \chi \in 0,1 \qquad (1)$$

Dynamic programming

The Dynamic programming is an algorithm design technique for optimization problems; often it minimizes or maximizes the results. The sequential decision problems are solved by using dynamic programming method by considering the class of solution methods. In the proposed MUTUAL-BENEFIT approach, the dynamic programming method is used to find an optimal solution for each task of an application by dividing the application into simpler sub-tasks. It has been effectively proven in many areas of solving optimization problem within a reasonable computation time.

5. EXPERIMENTAL EVALUATION

The proposed MUTUAL-BENEFIT approach is compared with NTGO [4] and E-LHEFT algorithms to exemplify the performance improvement of the MUTUAL-BENEFIT approach. The experimental results are evaluated using the Sudoku solver mobile gaming application.

Experimental setup

The CloudSim tool demonstrates a MUTUAL-BENEFIT approach to execute the Sudoku solver application. The implementation of Sudoku solver application evaluates the performance of the proposed approach in terms of device energy, response time, application completion time, and provider's profit. It considers n x n Sudoku solver table with n^2 cells. The Sudoku solver has several conditions while filling digits 1… n in cells. Consider, the mobile device solves few puzzles in the n x n table, and the mobile device offloads the remaining cells based on the task complication. The simulation is conducted in various scenarios by varying the number of mobile user requests from 500 to 2500, the level of Sudoku in terms of 'n' from 3 to 25, and the filled cells from 20% to 40%. The resource rich cloud server is considered as heterogeneous that has different MIPS value represents processing speed. The proposed approach is taken into the account of 10-50 PM resources and 100-1000 VM resources. Each CPU has the various ranges of the energy consumption that depends on the utilization, processing time and load of the resource.

Evaluation metrics

Energy level: It is defined as the percentage of energy retained by the mobile device while executing the mobile application.

Response time: It is the interval between the service initiated a time of an application and service resulted in a time of that application by the cloud service provider.

Application completion time: It is the overall completion time of a mobile application during mobile execution, offloading, and cloud execution.

Profit: It is the percentage of attaining profit of the provider after providing the service to the end-user. The profit measurement includes response time and energy cost with the consideration of resource utilization.

Experimental results and analysis

This section discusses the performance improvement of the MUTUAL-BENEFIT with the comparison of NTGO, E-LHEFT, ACO, and EAPA approaches when evaluating the system for Sudoku solver application. It reveals the performance in terms of Application complexity level Vs Energy level, Number of requests Vs Response time, Application complexity level Vs Application completion time, and Number of requests Vs Profit.

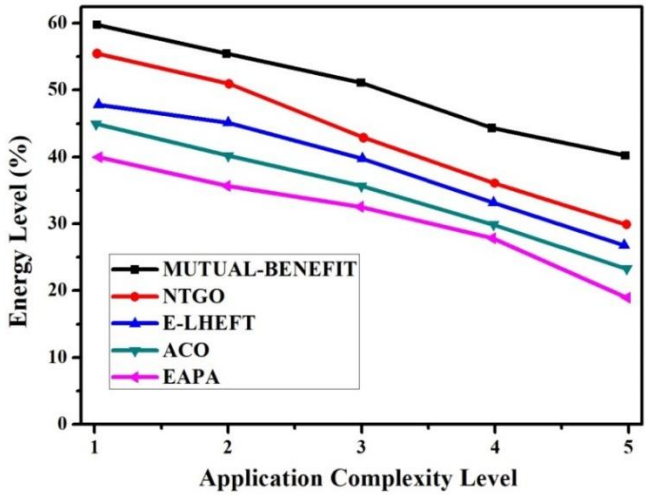

Figure 3: Application complexity level Vs Energy level

Figure 3 shows the percentage of energy levels on the mobile device while varying the complexity levels of the mobile application for the proposed MUTUAL-BENEFIT and the existing NTGO, E-LHEFT, ACO, and EAPA approaches with 2000 MIPS of VM resource. The graph indicates five complexity levels of Sudoku grid levels, such as 3×3, 6×6, 9×9, 16×16, and 25×25. The energy level percentage of the MUTUAL-BENEFIT approach and the existing comparative approaches linearly decreases while increasing the complexity level of Sudoku application from level 1 to level 5. Initially, in the MUTUAL-BENEFIT approach, the offloading manager of ThinkAir architecture effectively conserves the device energy, since it offloads the intensive tasks according to the device constraints. But the existing approaches suddenly drop energy level by 25% to 45% when varying the complexity levels from 1 to 5. In the same scenario, the MUTUAL -BENEFIT approach marginally decreases by 20% of the battery level. At the level 5, the MUTUAL-BENEFIT approach saves the device energy by 10% than existing approaches; since the proposed approach exploits the non-recursive 100 dynamic programming based ACO technique and parallel execution of tasks of an application. Table 1 represents the numeric values of Figure 3.

Table 1: Application complexity level Vs Energy level

Application complexity level	Energy level (%)				
	MUTUAL-BENEFIT	NTGO	E-LHEFT	ACO	EAPA
1	59	55	48	45	40
2	55	50	45	40	36
3	51	43	39	35	32
4	44	36	33	30	28
5	40	30	27	23	19

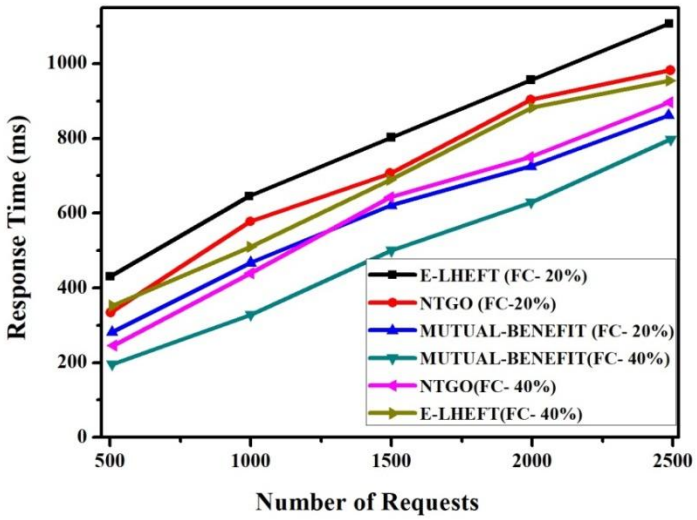

Figure 4: Number of requests Vs Response time

Figure 4 and Table 2 indicates the response time of the proposed MUTUAL-BENEFIT approach with the

existing NTGO and E-LHEFT approaches while increasing the number of requests submitted by the mobile users and the percentage of Filled Cells (FCs). The percentage of FC is referred as the ratio of the number of filling cells in the total number of cells of Sudoku solver application. The experimental evaluation of Figure.4 shows the variation of response time when FC=20% and FC=40%. The response time escalates while increasing the number of requests for the similar application. The performance of the MUTUAL-BENEFIT approach is higher than the NTGO approach after reaching 1000 number of requests; even the filled cells of the NTGO approach are higher than the MUTUAL-BENEFIT approach. This performance improvement is achieved by exploiting the non-recursive dynamic programming assisted ACO based effective task scheduling of an application. Also, the ThinkAir architecture based intensive application offloading method nearly reduces the unbearable delay of the application processing. But, the response time of NTGO approach suddenly escalates by 40%, while varying the number of requests from 1500 to 2500 with FC=20%. In the same scenario, MUTUAL-BENEFIT approach marginally increases by 37.7%, while using ACO with dynamic programming instead of using ACO with brute-force searching method. The E-LHEFT algorithm performs closer to the MUTUAL-BENEFIT in providing the response to the end-users. The E-LHEFT and MUTUAL-BENEFIT obtains a 605ms increase in response time when the number of requests varies from 500 to 2500 when FC=40%.

Table 2: Number of requests Vs Response time

Number of requests	Response time (ms)					
	MUTUAL-BENEFIT		NTGO		E-LHEFT	
	FC=20%	FC=40%	FC=20%	FC=40%	FC=20%	FC=40%
500	281	197	336	248	430	352
1000	468	329	578	438	648	509
1500	621	500	705	642	802	689
2000	728	625	903	750	954	881
2500	861	798	982	894	1105	952

The comparative result of application completion time is shown in Figure 5 while varying the application complexity levels and the percentage of filled cells. The corresponding numeric values of Figure 5 are shown in Table 3. The proposed MUTUAL-BENEFIT and the existing NTGO and ELHEFT approaches slightly increase the overall application completion time with varying number of complexity levels. In MUTUAL-BENEFIT approach, the application completion time depends on the satisfaction of the SLA objectives which is achieved by optimal offloading using ThinkAir architecture, optimal task scheduling using non-recursive dynamic programming based ACO technique and allocate the cloud server resources based on Bellman's optimality principle. The performance in terms of application completion time of NTGO approach is extended by 20% from the MUTUAL-BENEFIT approach when the application complexity level=5 and FC=40%. When FC=40%, the performance of NTGO approach is nearly equal to the MUTUAL-BENEFIT approach in FC=20% of the points of 2 and 3 of application complexity levels since the proposed approach shortens the longer execution time of an application using load -aware task scheduling and parallel execution. The E-LHEFT algorithm provides the similar performance of the NTGO approach with the slight increase of 2.36% and 5.61% when FC=20% and FC=40% respectively, even when increasing the application complexity levels.The results and discussion may be combined into a common section or obtainable separately. They may also be broken into subsets with short, revealing captions.

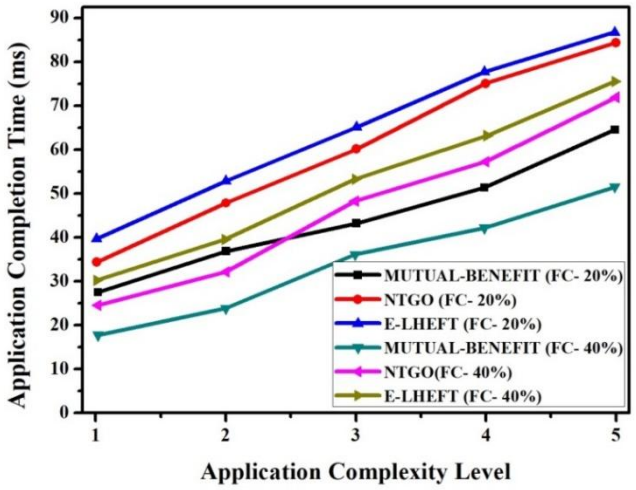

Figure 5: Application complexity level Vs Application completion time

Table 3: Application complexity level Vs Application completion time

Application complexity level	Application completion time(ms)					
	MUTUAL-BENEFIT		NTGO		E-LHEFT	
	FC=20%	FC=40%	FC=20%	FC=40%	FC=20%	FC=40%
1	27	18	34	25	40	30
2	37	24	48	32	53	39
3	43	36	60	48	65	53
4	51	42	75	57	78	63
5	64	51	84	71	87	75

Figure 6 depicts the profit of the cloud service provider while varying the number of requests and the percentage of filled cells. The corresponding numeric result values are tabulated in the Table 4.The experimental graph shows a slight variation in the profit for MUTUAL-BENEFIT, NTGO, and ELHEFT approaches. The experimental evaluation considers that the provider's maximum utilization level is completed when reaching 1500 mobile user's requests. Hence, the profit of the provider slightly increases until to reach the number of requests as 1500, after that profit gets a deviation from the peak point since the profit decreases when occurring over -utilization of the resources. However, the profit of the NTGO approach continuously decreases by 23.05% when increasing the number of requests from 500 to 2500 and FC=20%, because it allocates the resources without the knowledge of considering the trade-off between the SLA objectives and resource cost. In MUTUAL-BENEFIT approach, the provider's profit assignment is corresponding to the overall resource utilization and overall resource cost of the particular request processing on the server during optimal resource allocation. Thereby, the NTGO approach decreases the profit level to 22.7% more than that of MUTUAL-BENEFIT when the number of requests is 2500 and FC=40%, which reveals that the profit of NTGO approach gets unexpected deviation due to the absence of trade-off consideration.

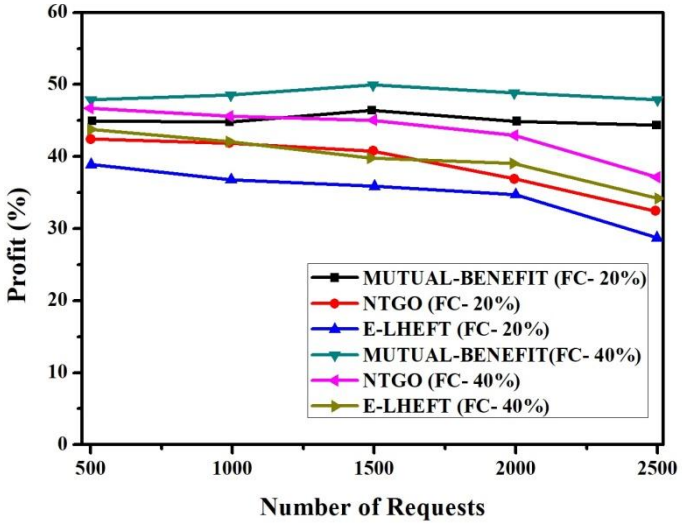

Figure 6: Number of requests Vs Profit

Table 4: Number of requests Vs Profit

Number of requests	Profit (%)					
	MUTUAL-BENEFIT		NTGO		E-LHEFT	
	FC=20%	FC=40%	FC=20%	FC=40%	FC=20%	FC=40%
500	45	48	42	47	39	44
1000	45	49	42	46	37	42
1500	46	50	41	45	36	40
2000	45	49	37	43	35	39
2500	44	48	32	37	29	34

6. CONCLUSION

In summary, the SLA-based optimization approach is presented to satisfy both the end users and service providers in MCC framework. The proposed MUTUAL-BENEFIT approach using optimal task offloading, task scheduling, resource selection, and provider selection for mobile application execution are explained clearly.

ISSN: 1674-8190

The main contribution of MUTUAL_BENEFIT approach is to provide the energy-efficient seamless mobile application execution without violating the SLAs. Moreover, it targets on maximizing the profit of the cloud service provider. In order to satisfy the SLAs, the MUTUAL-BENEFIT approach exploits the ThinkAir architecture which offloads the resource and compute intensive tasks to the cloud based on the energy model of the mobile device. The energy model based dynamic computation offloading prolongs the battery lifetime of the mobile device and provides the seamless mobile application execution. The MUTUAL-BENEFIT approach employs the enhanced dynamic programming based ACO method, which effectively schedules the intensive tasks with the consideration of objective function satisfaction. By utilizing the dynamic programming method along with the ACO technique facilitates the execution system in reducing the additional processing time of the recursive tasks. Finally, the MUTUAL-BENEFIT approach maintains the trade-off between the SLA objectives satisfaction and profit of the provider maximization by Bellman optimality principle and utility function based optimal resource allocation and provider selection. The utility function focuses on the resource utilization and resource cost while allocating the resources to the tasks scheduled by the dynamic programming based ACO method. Thus, the proposed algorithm retains the energy level in mobile devices by 10%, minimizes the response time by 12% and application completion time by 20%, and maximizes the profit of the cloud service provider by 11% for mobile applications.

7. ACKNOWLEDGEMENTS

The authors thank College of Arts and Sciences, Tanomah, King Khalid University, Saudi Arabia.

REFERENCES

[1] Feller, Eugen, Louis Rilling, and Christine Morin, “Energy-aware ant colony-based workload placement in clouds”, IEEE Computer Society in Proceedings of the 12thInternational Conference on Grid Computing, pp.26-33, 2011.

[2] Mishra, Ratan, and Anant Jaiswal, “Ant colony optimization: A solution of load balancing in cloud”, International Journal of Web and Semantic Technology, Vol.3, No.2, pp.33-50, 2012.

[3] Xue Lin, Yanzhi Wang, Qing Xie, and Massoud Pedram, “Energy and Performance-Aware task scheduling in mobile cloud computing Environment”, IEEE International Conference on Cloud Computing, pp.192-199, 2014.

[4] Wang Hinayana, Xue Lin, and Massoud Pedram, “A Nested two stage game-based optimization framework in mobile cloud computing system”, IEEE 7th International Symposium on Service Oriented System Engineering (SOSE), pp.494-502, 2013.

[5] Liu, Yanchen, and Myung J.Lee, “An effective dynamic programming offloading algorithm in Mobile cloud computing system”, IEEE transaction on Wireless Communications and Networking Conference (WCNC), pp.1868-1873, 2014.

[6] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang, “ThinkAir: Dynamic resource allocation and parallel execution in cloud for mobile code offloading”, IEEE Proceedings INFOCOM, pp.945-953, 2012.

[7] Kosta, Sokol, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang, “Unleashing the power of mobile cloud computing using ThinkAir”, arXiv preprint arXiv: 1105.3232, 2011.

[8] Atta Ur Rehman Khan, Mazliza Othman, Sajjad Ahmad Madani, and Samee Ullah Khan, “A survey of mobile cloud computing application models”, IEEE transaction on Communication Surveys, Vol.16, No.1, pp.393-413, 2013.