4. SLA-BASED OPTIMIZATION

4.1 INTRODUCTION

SLA is a crucial consideration of both the perspectives of the mobile end user and the cloud provider [137]. 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, this research work proposes Maximizing User antTicipation on cloUd-based mobile AppLications BesidE NEt proFIT (MUTUAL-BENEFIT) approach. 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.

4.2 AN OVERVIEW OF EXISTING OPTIMIZATION TECHNIQUES

ACO meta-heuristics [66] dynamically schedules the workload based on the current workload and resource availability. ACO based load balancing [68] 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 [103] 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 [89] minimizes the device energy and improves the performance of response time by effective offloading decision, and also increases provider’s profit. DPOA [95] 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.

4.3 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 $i = \{1, 2, \ldots, m\}$, and cloud resources $j = \{1, 2, \ldots, n\}$. An appropriate assignment of $j \in$ 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 $(\tau_{ij})$, load balancing $(\eta_{ij}(S_w'(t)))$, and profit $(S_w(t))$ in which $\eta_{ij}$ 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 $A_i$, 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.

**4.4 MUTUAL-BENEFIT METHODOLOGY**

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.

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 4.1.
4.4.1 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 [144] [145] [154] to make the offloading decision on the mobile cloud environment dynamically. Figure 4.2 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.

![Diagram of ThinkAir framework](image)

Figure 4.2: Architecture of ThinkAir framework [154]

4.4.1.1 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. Client handler executes the communication protocol, manages the connection from mobile users, receive and execute outsourced code and return the results. The ThinkAir architecture enables the profiler to monitor the mobile device and retains the information for future offloading decisions. The profiler component of ThinkAir architecture monitors the profile information of the device (CPU resource utilization, screen brightness), program (number of completed tasks, execution time of each task), and network (data transfer rate, number of tasks transmission and reception rate) of the mobile device to identify the off-loadable tasks. The energy estimation model (ThinkAir energy profiler) dynamically estimates the energy consumption of each running methods. ThinkAir components on the server side are the application server, client handler, execution controller, dynamic object input stream, and application profiler. The application server handles the offloaded methods. During the processing of offloaded methods, the exceptions are controlled by dynamic object input stream component.

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.

4.4.2 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 tasks using ACO technique, the MUTUAL-BENEFIT obtains the cloud tasks of an application from the offloading manager.

### 4.4.2.1 Service level agreements

The MUTUAL-BENEFIT approach takes into the account of SLAs during task scheduling and resource allocation. SLA is the agreement between the end-user and the cloud service provider. The proposed approach considers the SLAs as in satisfying the user’s desired requirements and providing the QoS by the CSP. QoS refers that the optimal cloud service provision without compromising the provider side. The proposed SLA parameter for mobile users ensures high-level user satisfaction in terms of battery power energy saving, minimum response time, and minimum completion time. The SLA parameter for cloud service provider enhances the profit of the service provider in terms of increasing resource utilization, and remote server energy cost. The mobile application execution in the cloud environment makes an impact on the profit of the provider and the SLA objectives. Since, if the cloud manager extends the processing time of an application than average execution time, an SLA violation occurs in terms of battery usage and response time.

### 4.4.2.2 ACO technique
The ACO technique is a probabilistic technique to solve computational infeasible problems. In ACO, the ants deposit pheromones on the ground that forms a trail. The trail attracts other ants. The pheromones evaporate faster on longer paths. More pheromone on path increases the probability of path being followed. The shortest route is found using pheromone trails which ants deposit whenever they travel, as a form of indirect communications. Thus, the ACO algorithm has been used to produce near-optimal solutions. The technique can run continuously to adopt the changes in real time.

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 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’s pheromone value based on the satisfaction of SLA objectives. The pheromone value updation depends on the optimal solution of an ant while mapping task into resource at a time $(t)$. $\chi$ 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 4.1.

$$\tau_i(T') = (1-\chi)\tau_i(t) + \Delta\tau_i(t,T')$$, where $\chi \in [0,1]$ (4.1)

A partial solution of each ant varies the current pheromone value of tasks over time $(t)$ to obtain the objective function. Hence, pheromone value increment completely depends on the partial solution of the ants $(k)$ that is tasks of an application. Ant’s partial solution based incremental pheromone value at updated time is shown in equation 4.2.
In equation 4.2, $\Delta T_i(t,T')$ represents the partial solution of the task that is obtained by $k^{th}$ ant at updated time $T'$. Each task has various probability values while mapping the same task to different resources.

4.4.2.3 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. The collection of cost value for each solution method is maintained in the computational cost structure. The optimal substructure varies across different problem domains. Mainly, the dynamic programming reduces the computation by solving subproblems in a bottom-up fashion, storing solution to a subproblem when it is solved the first time and analyzes the solution method when the subproblem has occurred again, and also select the solution method with less computational cost value. The solution to one subproblem which affects the solutions to other subproblems. The dynamic programming method ensures a low time complexity while executing the application to preserve the device energy and mitigate the application completion time.

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.

4.4.2.3.1 Non-recursive dynamic programming in ACO

The MUTUAL-BENEFIT exploits the non-recursive dynamic programming in ACO algorithm to reduce the computation complexity. It considers together of execution time, load balancing, and profit as ACO
parameters or pheromone value to achieve SLA objectives. The proposed technique monitors the remote process and retains the information for scheduling decision in future. If the system receives the recursive application, it selects the processes required to execute the recursive application of the previous remote process of the same application. Moreover, it merely depends on the previous storage tasks, but not on its results. Because the dynamic programming based previous storage result is not suitable for all recursion of tasks due to a different configuration of similar applications from multiple users.

For instance, image processing application includes a various process such as preprocessing, filtering, and recognition. The MUTUAL-BENEFIT algorithm schedules the recursive task to the previously scheduled VM, if that VM is available at the specific time. And also the proposed approach takes only the process, not an exact result of previous tasks while determining similar kinds of tasks. It reduces the iterative method of best VM resource selection when recursive task occurs. Hence, the time complexity is reduced when executing the resource intensive applications using MUTUAL-BENEFIT approach.

### 4.4.2.3.1.1 Objective function

The MUTUAL-BENEFIT approach ensures objective function that minimizes the completion time, balances the load, and maximizes the provider's profit during task-resource mapping in the mobile cloud environment. The equation 4.3 calculates the estimated completion time of tasks in cloud resources.

\[
T_i(ECT) = \frac{\text{Total length of task}}{VM_{Pe} \cdot VM_{MIPS}} + \frac{\text{Input file size}}{VM_{BW}}
\]  

(4.3)

To find the estimated load of each virtual machine, MUTUAL-BENEFIT approach exploits the following equation 4.4.

\[
L_j = L_j' + \frac{C_j^n}{C_j}
\]  

(4.4)
Where,

\[ L_i = \frac{\sum_{n=1}^{n} C_{ij}^n \lambda_{ij}}{C_i^n} \]

The profit of the service provider is based on overall resource utilization and overall resource cost. The equation 4.5 shows the profit calculation in cloud server provider.

\[
\text{Profit} = (\text{Overall utilization rate} \times \text{Overall response time}) - (\text{Unit energy cost} \times \text{Energy consumption})
\]

(4.5)

The MUTUAL-BENEFIT approach focuses on the optimization problem. Accordingly, the goal of MUTUAL BENEFIT approach can be expressed in the following equation 4.6 & 4.7

\[
\begin{align*}
\text{Minimize} & \sum_{i=1}^{m} E_{M}^i \lambda_{ij} \text{ and } \sum_{i=1}^{m} RT^i \lambda_{ij} \\
\text{Maximize} & \sum_{j=1}^{n} P_j \lambda_{ij}
\end{align*}
\]

(4.6) (4.7)

Subject to the constraints:

\[
\sum_{i=1}^{m} C_{ij} \lambda_{ij} \leq C_j \text{, Where } 0 \leq \lambda_{ij} \leq 1
\]

Where, \(E_M\), \(RT\), and \(P_j\) refers the energy consumption of the mobile device, response time of \(i^{th}\) mobile application, and profit of \(j^{th}\) provider respectively while executing the mobile application. \(\lambda_{ij}\) is the random parameter. \(C_{ij}\) and \(C_j\) represent the resource consumed by \(i^{th}\) task and total capability of the resource. Mobile device battery lifetime, response time, and profit of the service provider are the main consideration attributes during resource allocation in which device energy and response time are considered as the SLA objectives in the end-user perspective and profit as the SLA
objective in the provider perspective. Hence, this proposed approach is a target to achieve the aforementioned objective function.

4.4.2.3.1.2 Pheromone value calculation

The pheromone value is the probability value of tasks on cloud resources based on satisfying the objective function. The pheromone value of tasks in each VM is measured to achieve the optimal scheduling in MUTUAL-BENEFIT. Consider, the set \((S_i)\) contains the cloud tasks, in which ‘i’ is varied from 1 to the number of cloud tasks. The Pheromone Value (PV) of each task is measured using the following equation (4.8),

\[
PV_{ij} = \begin{cases} 
\frac{\alpha (r_{ij}(t)) [S_{\omega(t)}(t)_{\text{best}}]^{\beta}}{\arg\max_{n \in \text{allowed}_\omega(t)} [\alpha (r_{in}(t))] [S_{\omega(t)}(t)_{\text{best}}]^{\beta}}, & \text{if } T_i=T_1 \\
\frac{\alpha (r_{ij}(t)) [(\eta_{ij}) S_{\omega(t)}(t)]^{\beta} [S_{\omega(t)}(t)_{\text{best}}]^{\gamma}}{\arg\max_{n \in \text{allowed}_\omega(t)} [\alpha (r_{in}(t))] [(\eta_{in}) S_{\omega(t)}(t)]^{\beta} [S_{\omega(t)}(t)_{\text{best}}]^{\gamma}}, & \text{Otherwise}
\end{cases}
\] (4.8)

Where ‘i’ represents the task, and ‘j’ represents the VM and \(S_{\omega(t)}(t)_{\text{best}}\) is the best profit provider who satisfies the SLA objectives. \(T_{ij}(t)\) is the completion time of \(i^{th}\) task on \(j^{th}\) VM resource. \(T_{in}(t)\) is the completion time of \(i^{th}\) task \(n^{th}\) PM resource, where \(jen\). \(S_{\omega(t)}(t)\) is the optimal load balancing factor which balances the application execution using \(\eta_{ij}\). \(\eta_{ij} S_{\omega(t)}(t)\) is the load balancing value of \(i^{th}\) task on \(j^{th}\) VM resource. \(\eta_{in} S_{\omega(t)}(t)\) is the load balancing value of \(i^{th}\) task \(n^{th}\) PM resource. ‘allowed\(_{\omega(t)}\)’ denotes the set of feasible solutions in terms of VMs. \(\alpha, \beta, \text{ and } \gamma\) parameters represent the weight of each term in \(PV_{ij}\), are used to control the relative importance of each factor. (i.e.\(0 \leq \alpha, \beta, \gamma \leq 1\). The ACO technique applies this equation for all tasks in all VMs, and it selects best VM with maximum \(PV_{ij}\) for each task using the equation 4.9.

\[
VM_{\text{best}} = \begin{cases} 
\arg\max_{n \in \text{allowed}_\omega(t)} [\alpha (r_{in}(t))] [(\eta_{in}) S_{\omega(t)}(t)]^{\beta} [S_{\omega(t)}(t)_{\text{best}}]^{\gamma}, & \forall R_j=C_{ij} \\
0, & \text{Otherwise}
\end{cases}
\] (4.9)
Each task has various $PV_{ij}$ value while mapping with different resources. In equation 4.8, if a task ($T_i$) is the initial task ($T_1$) of an application and the server, it selects the resource based on only the weight of estimated completion time and profit. Then, it constructs the remaining schedulable task list and selects the optimal resource for each task besides considering load balancing of VM resource. This process facilitates the system to execute and complete the consecutive tasks of an application. For example, there are four faces in an image and if the first two phases are scheduled in VM with high processing speed, but others in a VM of less processing speed, this leads to delay the application completion time due to the lack of load balancing among tasks in the application.

Even though the non-recursive dynamic programming ACO technique is sufficient to satisfy the SLA objectives and the profit of the provider, it is necessary to decide the optimal value of $S_{w(t)}$ to maintain the best trade-off between SLA objectives satisfaction and profit of the provider in the optimal VM selection.

4.4.3 Enhancing profit for cloud provider via optimal resource allocation

SLA objectives and provider’s profit are interrelated with each other. If any violations occur in SLA objectives, it may affect the profit of the cloud provider in terms of increasing resource cost. Bellman’s theory is used to allocate the optimal VM resources based on the resource utilization (initial state) and resource cost (initial decision) of each task. The remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. With the aim of maximizing the cloud service provider’s profit, the MUTUAL-BENEFIT approach is targeted to achieve the load balancing of an application that also provides the long-lasting device battery. The consistent application execution on remote server reduces the energy cost that improves the response time and profit of the system. The profit of the provider is not static and is varied dynamically according to the cloud service provisioning by the service provider.
4.4.3.1 Resource selection

The consideration in determining the VM resource utilization is crucial to increase the kth provider’s profit while mapping the tasks to the VM resources. It balances the load of the tasks to achieve the target of the overall completion time reduction of an application using Bellman’s theory which is based on the dynamic programming method by focusing initial and remaining optimal solutions. This dynamic programming method divides the decision problem of optimal solution into the subproblems of initial and future solutions. For instance, if the system allocates the VM to the initial task at 0.3 resource cost, the proposed approach attempts to allocate the second task on that same VM when there is no VM availability with the cost less than 0.3. Otherwise, it selects the VM with the various cost which is proximity to 0.3. This every optimal solution is used to calculate the final solution in terms of profit, which is similar to the user satisfaction level through response time.

In equation 4.10, $U(i)$ and cost represent the resource utilization and resource cost respectively. The satisfaction of the mobile user is based on the utility function that measures service performance while meeting user’s anticipation in terms of response time. Utility function measures the user’s satisfaction based on the response time of each task and the utilization of resources for completing the application execution.

$$S_{\omega}(t)_k = \sum_{i=1}^{m} \sigma_i \cdot U(i) \left( \sum_{j=1}^{n} \Phi_{ij} \left( RT_{ij}^c + RT_{ij}^e \right) \right) - \text{cost} \left( \sum_{j=1}^{n} \left( \mu P_{ij}^\text{con} + \nu \left( P_{ij}^e \right) \left( \Psi_{ij}^e \right) \right) \right)$$  (4.10)

Where $\sigma_i$ is the arrival rate of the ith request, and $\Phi_{ij}$ is the task assigned a probability of jth VM resource. $RT_{ij}^c$ and $RT_{ij}^e$ are the service response time of the ith task in which superscript ‘c’ and ‘e’ represents at the stage of communication and execution (computation). $\mu$ and $\nu$ parameter denotes the server ON (1) or OFF (0) state and allocated (1) or not (0) respectively. $P_{ij}^\text{con}$ and $P_{ij}^e$ are the constant energy consumption of server and consumed energy in the execution stage. $\Psi_{ij}^e$ is the portion of jth allocated server resources to ith task request execution.
4.4.3.2 Provider selection

The selection of the best service provider is based on the response time and resource cost. According to Bellman’s equation, it formulates the evaluation function for allocating the resource to the selected task. This resource allocation considers the initial solution regarding utilization of each task and resources cost for executing the entire application. The relationship between the service response time and service cost finds the user satisfaction level. The utility function represents the measurement of usefulness for both the mobile user (response time) and the cloud service provider (resource utilization and resource cost) while executing the mobile applications.

Utility function based allocation focuses on two factors such as user’s anticipations and profit of the provider. The profit of the service provider varies according to the fluctuation of response time and cost. The service provider charges high when the user receives the immediate response from the provider. Therefore, the service provider is targeting to achieve a high level of user satisfaction and also its profit \( S_{\omega}(t)_{\text{best}} \). The optimal profit value based on the following profit calculation of the system is given in equation 4.11,

\[
S_{\omega}(t)_{\text{best}} = \underset{k \in S_p}{\arg \max} [S_{\omega}(t)_{k}]
\]  

The proposed algorithm of MUTUAL-BENEFIT is shown in Algorithm 4.1. In Algorithm 4.1, \( T_{SS} \) indicates the Selected Schedulable Task, and \( \xi(S(t)) \) represents the satisfaction of objective function based best VM resource to the corresponding task. \( \Delta T_{t}^{i}(t,T') \) is the incremental update of a partial solution of the \( i^{th} \) task at time \( T' \) and \( R_{t}^{A} \) is the allocated VM resource which satisfies both the factors of \( S_{\omega}'(t) \) and \( S_{\omega}(t)_{\text{best}} \). Finally, the proposed approach uses the ThinkAir architecture to execute the allocated VMs in a parallel manner. Moreover, it optimizes the better trade-off between user satisfaction and profit of the system.
Algorithm 4.1: Algorithm of MUTUAL-BENEFIT in MCC

Input: Mobile applications and cloud resources
Output: Optimal task offloading, task scheduling, resource allocation, provider selection

//Initialization
Step 1: Initialize tasks (T_i), resources (R_j), and pheromone value of ant(k)
   Optimal solution=0; profit=0; load=0
Step 2: Check set of feasible solutions at time ‘t’ (allowed_ω(t))≠0
Step 3: Assign i=0, check i<m, increment ‘i’ value
Step 4: Assign j=0, check j<n, increment ‘j’ value
Step 5: Check (T_i = T_1)
   Select schedulable tasks T_s and cloud resources R_j based on time, profit
   Otherwise select schedulable tasks T_s and cloud resources R_j based on time, profit, load

//Finding an objective based solution
Step 6: Find E_i^C, E_i^P, and E_i^L of all resources
Step 7: Calculate optimal solution
   Optimal solution= {arg min(E_i^C), arg max(E_i^P), arg min(E_i^L)}
   η_ij = min(E_i^C)
Step 8: Sort R_j and T_i based on E_i^C, E_i^P, and E_i^L in ascending order

//Selection of schedulable task and resource
Step 9: Check (T_i = T_{SS})
   Find current optimal solution ξ(S(t)) = [α(Τ_i(t))] [(η_ij)S_ω′(t)]^β [S_ω(t)_{best}]^γ
Step 10: Schedule T_i based on optimal solution
   Add selected schedulable task T_{SS} to optimal load balancing factor S_ω′(t), best service provider S_ω(t)_{best}, and tabu list
Step 11: Update completion time of i^{th} task on j^{th} VM resource Τ_i(t) on pheromone table

//Calculating optimal solution
Step 12: Calculate Δ_{T_i}^ω (t,T′) using ACO
Step 13: Check previous optimal solution with current optimal solution
   (ξ′(S(t))<ξ(S(t)))
Assign optimal solution=ξ(S(t))
Update final optimal solution to provider

Step 14: Allocate R_j based on ξ(S(t))
Step 15: R_j^A satisfies S_ω'(t) and S_ω(t)_{best}
Calculate profit of the system

4.4.3.2.1 Optimal solution

The utility function measures the optimal solution for best service provider selection by considering the user’s satisfaction on the response time of each task and the utilization of resources for completing the application execution. The equation 4.12 calculates the optimal solution using the utility function principle among heterogeneous cloud service providers.

Optimal solution= \max_{i \in A} S_ω(t)_{best} \{(R_C,R_T)_0+\mu(R_C,R_T)_n\} , for all j' (4.12)

Where \sigma_i is the S_ω(t)_{best} is the best service provider, (R_C,R_T)_0 is the resource cost (R_C) and response time (R_T) of initial solution for task of an application, and (R_C,R_T)_n is resource cost and response time of remaining tasks of an application measured from the initial solution and initial task.

4.5 EXPERIMENTAL EVALUATION

The proposed MUTUAL-BENEFIT approach is compared with NTGO [89] 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.

4.5.1 Experimental setup

The CloudSim tool demonstrates an 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 to 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.

4.5.1.1 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.

4.5.2 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.
4.5.2.1 Application complexity level Vs Energy level

Figure 4.3: Application complexity level Vs Energy level

Figure 4.3 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
dynamic programming based ACO technique and parallel execution of tasks of an application. Table 4.1 represents the numeric values of Figure 4.3.

Table 4.1: Application complexity level Vs Energy level

<table>
<thead>
<tr>
<th>Application complexity level</th>
<th>MUTUAL-BENEFIT</th>
<th>NTGO</th>
<th>E-LHEFT</th>
<th>ACO</th>
<th>EAPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60</td>
<td>55</td>
<td>48</td>
<td>45</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>55</td>
<td>51</td>
<td>45</td>
<td>40</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>51</td>
<td>43</td>
<td>40</td>
<td>36</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>44</td>
<td>36</td>
<td>33</td>
<td>30</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td>30</td>
<td>27</td>
<td>23</td>
<td>19</td>
</tr>
</tbody>
</table>

4.5.2.2 Number of requests Vs Response time

![Figure 4.4: Number of requests Vs Response time](image)

Figure 4.4 and Table 4.2 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 application complexity level.
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 4.4 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 38.7%, 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. In essence, the E-LHEFT and MUTUAL-BENEFIT obtains a 600ms increase in response time when the number of requests varies from 500 to 2500 when FC=40%.

Table 4.2: Number of requests Vs Response time

<table>
<thead>
<tr>
<th>Number of requests</th>
<th>Response time (ms)</th>
<th>MUTUAL-BENEFIT</th>
<th>NTGO</th>
<th>E-LHEFT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FC=20%</td>
<td>FC=40%</td>
<td>FC=20%</td>
<td>FC=40%</td>
</tr>
<tr>
<td>500</td>
<td>280</td>
<td>200</td>
<td>340</td>
<td>250</td>
</tr>
<tr>
<td>1000</td>
<td>470</td>
<td>330</td>
<td>580</td>
<td>440</td>
</tr>
<tr>
<td>1500</td>
<td>620</td>
<td>500</td>
<td>700</td>
<td>640</td>
</tr>
<tr>
<td>2000</td>
<td>730</td>
<td>620</td>
<td>900</td>
<td>750</td>
</tr>
<tr>
<td>2500</td>
<td>860</td>
<td>800</td>
<td>980</td>
<td>890</td>
</tr>
</tbody>
</table>
4.5.2.3 Application complexity level Vs Application completion time

The comparative result of application completion time is shown in Figure 4.5 while varying the application complexity levels and the percentage of filled cells. The corresponding numeric values of Figure 4.5 are shown in Table 4.3. The proposed MUTUAL-BENEFIT and the existing NTGO and E-LHEFT 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 2.38% and 5.63% when FC=20% and FC=40% respectively, even when increasing the application complexity levels.
Figure 4.5: Application complexity level Vs Application completion time

Table 4.3: Application complexity level Vs Application completion time

<table>
<thead>
<tr>
<th>Application complexity level</th>
<th>Application completion time(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MUTUAL-BENEFIT</td>
</tr>
<tr>
<td></td>
<td>FC=20%</td>
</tr>
<tr>
<td>1</td>
<td>28</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
</tr>
<tr>
<td>3</td>
<td>43</td>
</tr>
<tr>
<td>4</td>
<td>51</td>
</tr>
<tr>
<td>5</td>
<td>64</td>
</tr>
</tbody>
</table>
4.5.2.4 Number of requests Vs Profit

Figure 4.6: Number of requests Vs Profit

<table>
<thead>
<tr>
<th>Number of requests</th>
<th>Profit (%)</th>
<th>MUTUAL-BENEFIT (FC=20%)</th>
<th>NTGO (FC=20%)</th>
<th>E-LHEFT (FC=20%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FC=40%</td>
<td>FC=40%</td>
<td>FC=40%</td>
<td>FC=40%</td>
</tr>
<tr>
<td>500</td>
<td>45</td>
<td>48</td>
<td>43</td>
<td>47</td>
</tr>
<tr>
<td>1000</td>
<td>45</td>
<td>49</td>
<td>42</td>
<td>46</td>
</tr>
<tr>
<td>1500</td>
<td>46</td>
<td>50</td>
<td>41</td>
<td>45</td>
</tr>
<tr>
<td>2000</td>
<td>45</td>
<td>49</td>
<td>37</td>
<td>43</td>
</tr>
<tr>
<td>2500</td>
<td>44</td>
<td>48</td>
<td>33</td>
<td>37</td>
</tr>
</tbody>
</table>

Figure 4.6 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 4.4. The experimental graph shows a slight variation in the profit for MUTUAL-BENEFIT, NTGO, and E-LHEFT 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 23.25% 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 22.9% 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.

4.6 SUMMARY

- In this chapter, 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.
- 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.

Chapter 5 discusses an adaptive workflow management mechanism for an energy-efficient IoT-based mobile application in mobile cloud IoT environment.