Computational offloading framework using caching and cloud service selection in mobile cloud computing

K. Sindhu* and H.S. Guruprasad

Department of ISE,
BMS College of Engineering,
Bangalore, India
Affiliated to: Visvesvaraya Technological University, India
Email: sind19@gmail.com
Email: drhsguru@gmail.com
*Corresponding author

Abstract: Execution of resource constrained applications on mobile devices is still a challenging task due to limited resources of mobile devices like processing speed, battery-power and network bandwidth. Mobile cloud computing enables mobile devices to execute the resource intensive tasks with the help of cloud servers. In this paper, we have implemented a computational offloading framework to offload resource intensive task of an application onto the cloud server. A caching scheme is proposed to further reduce the latency of execution and battery-power consumption of mobile devices. Multi criteria decision analysis methods are used to select the optimal cloud considering four cloud servers located at different regions. The results indicate a performance increase of 99% in execution time and 97% in energy consumption using the caching scheme and 92% in execution time and 90% in energy consumption using offloading on cloud server compared to execution on mobile device for the proposed application.

Keywords: mobile; cloud service selection; TOPSIS; offloading; mobile cloud computing; MCC; energy consumption; caching; analytic hierarchy process; AHP; multi-criteria decision analysis; MCDA; mobile device.

Reference to this paper should be made as follows: Sindhu, K. and Guruprasad, H.S. (2022) 'Computational offloading framework using caching and cloud service selection in mobile cloud computing', *Int. J. Advanced Intelligence Paradigms*, Vol. 21, Nos. 3/4, pp.189–210.

Biographical notes: K. Sindhu is currently working as an Assistant Professor in the Department of Information Science and Engineering at BMS College of Engineering, Bangalore, India. She received her MTech in Computer Network Engineering from the Visvesvaraya Technological University. She is currently pursuing her PhD at the Visvesvaraya Technological University. Her research interest includes cloud computing, mobile cloud computing, internet of things and mobile-based application development.

H.S. Guruprasad holds a PhD in Computer Science. He is working as a Professor in the Department of Information Science and Engineering at BMS College of Engineering, Bangalore, India. He has over two decades of experience in the teaching field. His research interests include networks and communication, cloud computing and internet of things.

1 Introduction

Mobile device faces various challenges such as shortage of processing power, inadequate storage and reduced battery life. With the advancement of cellular network, it is easier to incorporate cloud computing on mobile systems. To overcome the resource constraints of mobile devices, cloud computing is integrated with mobile computing which has paved way to a new terminology called mobile cloud computing (MCC). MCC is a paradigm consisting of mobile computing, cloud computing and communication network. MCC overcomes the limitation of mobile devices such as limited battery life and processing capabilities by augmenting the capability of cloud computing (Othman et al., 2013). Key objective of MCC is to boost the processing power and reduce the battery consumption of mobile devices by offloading computation intensive task onto the cloud data centres (Arun and Prabu, 2017; Somula and Sasikala, 2018; Fernando et al., 2013). Computation offloading boosts the capabilities of mobile devices by delegating the resource demanding task onto the cloud server.

Offloading is a promising method to solve the limitations of mobile devices where the idea is to migrate the computationally intensive task to the cloud servers and obtain the results. The decision to offload to cloud server or execute on mobile device is considered based on various decision criteria since the cost of offloading must be lesser than executing on the mobile device (Wu, 2018). Cloud computing facilitates end users in accessing computing resources from anywhere, anytime and pay as per the usage. MCC is an infrastructure where both the data processing and/or data storage happens outside the mobile device thereby facilitating a broad range of mobile subscribers to benefit from the usage of mobile cloud applications (Akherfi et al., 2018; Dinh et al., 2013; Noor et al., 2018; Al-Janabi et al., 2017).

Mobile device is a primary human need in today's information technology world. Resource intensive application execution on mobile phones leads to faster depletion of battery power. Execution of complex applications on mobile device is still a challenge due to resource constraint issues of mobile devices. Hence, the main objective of the proposed work is to enhance the battery life of mobile devices and faster execution of the application.

The objective is achieved by an offloading framework which offloads resource intensive task of the application onto the cloud server along with a caching scheme implemented on the client device. Network latency in mobile apps has a greater effect on user experience. Users expect the apps to be faster and responsive. Prefetching and caching are effective in reducing latency and energy consumption of mobile application. Caching helps in storing the data so that future requests are accessed more efficiently. It eliminates the need for a network request call if requested data is available in cache.

The work also addresses the issue of selecting the optimal cloud server for offloading the task. Cloud service selection considering mobile environment is not explored to the fullest even though there are many previous works in 'cloud service selection' domain. Fuzzy techniques were used in the previous works where linguistic variables were considered to represent the significance of each criterion and four cloud servers were assumed to be present. In the proposed work, four real time cloud servers located in different regions were considered; analytic hierarchy process (AHP) and technique for order preference by similarity to ideal solution (TOPSIS) were proved to be the viable solutions in finding optimal cloud server by comparing the theoretical and real time results obtained. The results indicate performance improvement of the mobile device

when the task is offloaded to the cloud server and further efficiency of mobile device is achieved by using the caching scheme.

The contributions of this paper are as follows:

- The execution of applications on mobile devices increases the latency and power consumption. Hence, an offloading strategy is proposed where cloud servers are utilised for executing the computation intensive task.
- Caching mechanism is introduced in this approach to further reduce the execution time and energy consumption. The result of the computation is cached so that next time the same request arrives from the end user, the cache is accessed in fetching the response. The network calls to the cloud server are reduced by incorporating caching scheme along with offloading and is useful when communication network is not available.
- Selection of optimal cloud server to offload computationally intensive task of the application is proposed considering four cloud servers located at different region. The real time testbed results are considered for ranking the cloud server.

The novelty of the proposed framework is to enhance the performance of the mobile device by choosing an optimal cloud server for offloading from a set of cloud servers along with a caching mechanism. The rest of the paper is organised as follows: Section 2 of the paper examines the research work done by other researchers. Section 3 demonstrates the proposed work followed by cloud path selection in Section 4. Section 5 represents the results and discussion followed by conclusions in Section 6.

2 Related work

The related work carried out in the field of computation offloading, caching and multi-criteria decision analysis (MCDA) is discussed in this section.

2.1 MCC infrastructure for offloading

In computational offloading, resource extensive computations are transferred from mobile device to the resource rich cloud server or cloudlets. The main objective of computational offloading (Enzai and Tang, 2014; Magurawalage et al., 2014) was to save mobile device energy and decrease execution latency. The mobile device battery consumption is minimised by offloading computationally intensive task onto the cloud servers. Computational offloading involves communication among mobile device and cloud server. Hence, communication cost involved must be considered when making an offloading decision which involves transmission delay/s, network bandwidth and energy. Offloading is ideal only when the cost of data transmission and execution is lesser than execution of the job on the mobile device. Cloud side offloading decision scheme (Jadad et al., 2018) is proposed to decide the task of execution on mobile device or cloud server. A comparison between latency of execution and energy consumption on mobile device and cloud server was conducted. The cloud side offloading scheme saves execution time and energy, but the network availability should be present to send the request to the cloud and find where the job needs to be executed. Saha and Hasan (2017) represent an offloading mechanism for execution of the job on cloud server considering previous

connection history to estimate the communication delay. Application used for the experiment was bubble sorting. Sarvabhatla et al. (2017) considers few mobile devices accessing the cloud resource to execute the task. The choice to offload or not to offload is based on cost tables considering metrics like size of data, time taken for execution, network bandwidth, etc. The cloudlet is used as the decision-making point to decide where to execute the task. Whaiduzzaman et al. (2015) uses cloudlet to augment the performance of the mobile device. The computationally intensive task is offloaded to the cloudlet hence the mobile device should be located closer to cloudlet for application execution. Liang et al. (2018) proposes cloudlet for offloading computation intensive task and cloud server was considered in case of unavailability of cloudlets. Considering the mobility of the device, two cloudlets are considered in their work. Thanapal and Durai (2018) proposes an offloading framework based on resource utilisation history, cloud capacity, energy consumption and delay tolerance using CloudSim simulator. Application considered for their study was a random matrix multiplication program. A comparison study on execution time and energy consumption on mobile device and cloud server were carried out. The results prove that offloading to the cloud server is better than execution on mobile device. Alsubhi et al. (2020) proposes a framework for offloading the resource intensive task, the decision to offload or not was taken by decision engine based on the cost metric for execution on mobile device and cloud. The application considered for their work was counting the total number of words in a file. A comparison on execution time on three different mobile devices and cloud platform were discussed. The results indicate as the file size increases the execution on cloud platform is better.

2.2 Caching

Caching is useful to reduce network calls and fetch the data extremely fast. Based on storage area, there are two types of cache: memory cache and disk cache (Android, 2019). The memory cache is faster in getting the data but once the app terminates, the data is no longer available since it is stored in the memory of the application. In disk cache, the data is retained even after the app terminates. When the app is executed again after the termination, the earlier data stored in disk cache can be accessed. In the proposed work, disk cache is used since the data can be retained in the cache for further use when the application is executed again. A middleware solution (Zhao et al., 2016) for pre-caching to improve execution efficiency and reduce network costs is proposed. The approach is useful when the app is developed using HTML and Webkit. Dutta and Vandermeer (2017) represent a caching mechanism at OS level and the cache is available to all mobile applications. Response and object level caching approaches have been implemented. The response cache stores full HTTP responses. In object level caching, each object refers to a portion of HTTP response. The results indicate object caching reduces energy consumption to larger extent than response cache. Zhao et al. (2018) discusses a framework where the cached data is stored on the mobile SD card, prefetching and caching schemes are suggested while developing mobile apps. Thirty-three different categories of apps were considered to check whether it is beneficial to prefetch HTTP requests and cache the responses. The analysis indicates that prefetching and caching can be beneficial across many apps. Progressive webapps (Malavolta et al., 2020) provides users with offline first experience by storing resources and java script modules on browser in dedicated cache when accessed for the first time. This helps users to work offline. The results indicate that with populated cache, the

progressive webapps load faster but does not have much impact on difference in energy consumption when loaded with an empty or populated cache.

2.3 Cache replacement algorithms

Cache replacement algorithms are used to manage the information stored in the cache. When the cache is full, there is a need to replace old contents of the cache to make way for the new contents. This section provides few cache replacement strategies (Safavat et al., 2020).

2.3.1 First in first out replacement

The item cached first would be the item which is moved out first when the cache is full. It is one of the hassle-free strategies for content replacement.

2.3.2 Least frequently used replacement

Least frequently used item in cache is evicted when the cache is full. In this method, there is a need to track how frequently an item in the cache is used.

2.3.3 Time-aware least recent used replacement

The contents are time stamped to see how often the content is used. The timestamp is used to decide which content to be removed from the cache.

2.3.4 Adaptive replacement cache

Extensively used and recently used data are tracked and the removal history of both is used to change the data of the cache.

2.3.5 Least recently used replacement

Least used data of the cache is tracked. The content which is least used is the one that would be removed when the cache is full. This method uses the storage space effectively.

In the proposed approach, the least recently used (LRU) replacement algorithm is used for cache content replacement when the cache is full since this method uses the storage space effectively.

2.4 Multi-criteria decision analysis

MCDA (Odu, 2019) aims in choosing the optimal solution among several alternatives by evaluating multiple conflicting criteria. It provides mathematical solutions for choosing best alternative among several alternatives based on different criteria. The general steps involved in MCDA methods are to first determine the relevant criteria based on which the alternatives will be selected. Next step is to select a set of relevant alternatives from which finally a decision is made to choose the best alternative. A matrix is constructed based on the criteria and alternatives. The weights of each criterion are determined based on the importance of the criteria for a specific problem. The best alternative is selected among a set of alternatives by assigning performance value for each alternative by using

MCDA methods. Wu et al. (2012) used AHP and fuzzy TOPSIS in fuzzy environment to choose an optimal cloud for mobile cloud environment. Five criteria and four alternatives were considered for their work, the experiment was based on numerical analysis using linguistic values for the criteria. Goudarzi et al. (2017) discusses the usage of genetic algorithm to select the ideal alternative in a multisite environment while offloading the computationally intensive task in MCC. Singla and Kaushal (2015) presents the usage of fuzzy AHP to select the optimal cloud server from a class of cloud servers in a MCC environment.

In most of the previous works, the task is migrated on cloudlet or a single cloud server or a combination of private and public servers for offloading. In the proposed work, we have considered four cloud servers located at different regions and proposed a viable solution for cloud service selection. AHP and TOPSIS is used for cloud service selection since the response time in getting the results is extremely fast and is known for computational simplicity (Zhou et al., 2015). Six criteria and four alternatives are considered in the work and the same is explained in the following section. The mobile device energy consumption and execution latency is further reduced by incorporating a caching scheme.

3 Proposed work

The key contribution of the proposed work is an offloading framework comprising of caching and selection of an optimal cloud server from four cloud servers located in different regions using AHP and TOPSIS. AHP and TOPSIS, the viable solutions for optimal cloud path selection are proved by comparing theoretical analysis results with experimental analysis in Section 4. Parallel execution of independent modules of the application using multithreaded programming is incorporated for faster response.

In the proposed work, the application considered is to solve the mathematical puzzle Tower of Hanoi. Tower of Hanoi is solved by finding the number of disk moves required for a given number of disks considering three pegs and the rules. When the end user starts the mobile app, the number of the disks to be moved is read as input from the user. The mobile device checks whether the executed result of requested task is available in the cache. If found available, it is a cache hit and the result is displayed on the mobile device. If requested data is not available in the cache, a cache miss occurs, and the task needs to be executed. In the next step, the communication network availability needs to be checked. If either Wi-Fi or cellular network connection is available, a request is sent to one of the cloud servers based on the ranking of the cloud server to execute the task. In case of network unavailability, the mobile device is used to perform the required operation. After execution of the job, the result is displayed on the mobile device. When the job is executed on mobile device or cloud server, a different thread is used to check the availability of space in the cache. Having a parallel execution here results in faster response. If cache is not full, then the result is added to the cache database. If the size of the cache exceeds the maximum size, then the existing data from the cache must be removed and the new result needs to be added. To remove the existing data from the cache the LRU algorithm is implemented. The LRU data from the cache is identified and removed to pave way for the new data to be inserted into the cache. The algorithm of the proposed work is represented in Figure 1 and the graphical representation of the algorithm is given in Figure 2.

Figure 1 Proposed algorithm

Considerations:

M_d – mobile device

C_s – cloud sever

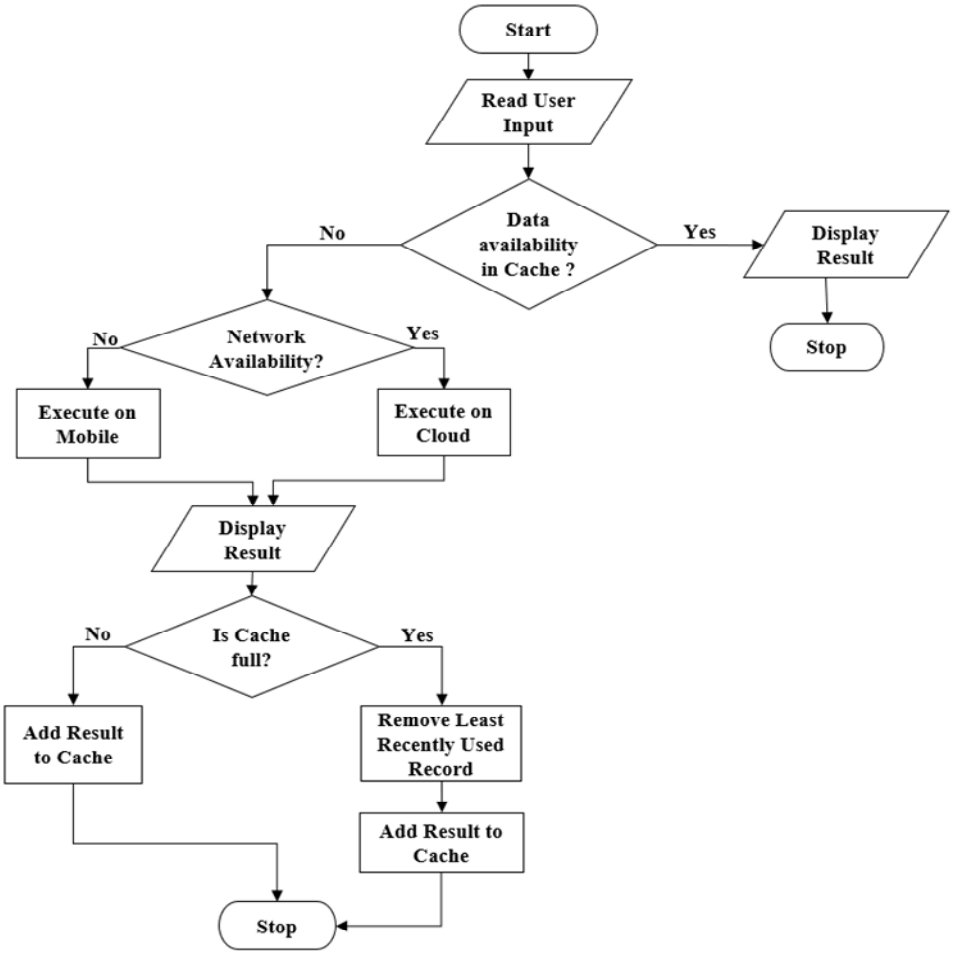
C_m – cache memory

C_{ms} – cache memory max. size

Procedure

```
Start
 Read the input of the task to execute from Md
 If data_available() in Cm
          Display the result on Md
          Update the count variable in Cm
 Else if check_network_availability()
      Cs ← select_cloudserver()
      Execute the task on Cs
      Cs sends the result back to Md
      Display the result on Md
       max_size ← maxsize_cache()
       If(max_size < Cms)
            Update() the new record in the Cm
       else
              Delete() the least recently used record in the Cm
              Update() the new record in the Cm
       End if
 Else
      Execute the task on Md
      Display the result on Md
      max_size ← maxsize_cache()
      If(max_size < Cms)
            Update() the new record in the Cm
      else
            Delete() the least recently used record in the Cm
            Update() the new record in the Cm
      End if
 End if
End
```

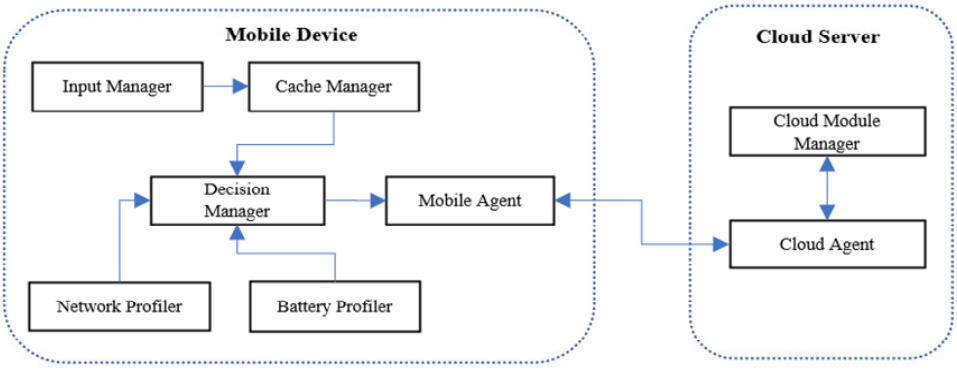
Figure 2 Flow diagram of the algorithm



The resource intensive job of the application is executed on cloud server on network availability else execution happens on mobile device. If any one of the cloud servers fails, the execution will be migrated to the next optimal cloud. MCDA techniques are used to find the optimal cloud from a class of four cloud servers positioned in different regions. The criteria and alternatives required for cloud path selection can be changed as per the developer requirement.

The cache memory is implemented in android using room library which provides an abstraction over SQLite for robust database access. The room library helps in creating a cache on a device that is running the app (Android, 2017). The structure of the cache memory contains the number of disks, the result, and the count variable for keeping track of the LRU item in the cache. Implementing the room database requires three components, the database holder, entity and data access objects.

Figure 3 Proposed architecture (see online version for colours)



The modules used in the proposed architecture are represented in Figure 3. At the mobile device end, six modules were implemented. The input manager is responsible to read the request from the mobile device user and communicate with the cache manager. The cache manager checks whether a cache hit, or a cache miss occurs and informs the decision manager. Cache manager module is also responsible to update the cache once the results are received from cloud agent or mobile agent and executes the LRU algorithm as per the requirement. Network profiler is responsible for checking the network availability for either the Wi-Fi network or the cellular network and sends the response to the decision manager. Battery profiler module monitors the battery status of the mobile device.

Based on the inputs received from cache manager, network profiler and battery profiler modules, the decision maker module decides where to execute the task. The decision manager is also responsible for choosing the ideal cloud. Based on the decision from the decision manager module, the mobile agent either communicates with the cloud server for job execution or execution happens on mobile device. The mobile agent uses the volley framework (Android, 2021) for communicating with the cloud agent. Volley is an HTTP library that makes network communication easier and provides faster communication. At the cloud end, the cloud agent is responsible for receiving requests and sending the response to the mobile agent. The cloud agent in turn communicates with the cloud module manager to execute the task and delivers the response back to the mobile agent. The experimental setup used for the proposed work is represented in Table 1.

Table 1 Device specifications

	Mobile device	*Cloud server 1*	*Cloud server 2*	*Cloud server 3*	*Cloud server 4*
Model	Samsung Galaxy Note 3 SM-N900	Amazon EC2 instance T2.Medium	Amazon EC2 instance T2.Medium	Amazon EC2 instance T2.Medium	Amazon EC2 instance C4.2xlarge
		North California	Asia Singapore	Asia Mumbai	Asia Mumbai
CPU	8 core, 1.90 GHz	2 vCPUs, 2.3 GHz	2 vCPUs, 2.3 GHz	2 vCPUs, 2.3 GHz	8 vCPUs, 2.9 GHz
RAM	3 GB	4 GiB	4 GiB	4 GiB	15 GiB

4 Cloud service selection

MCDA (Velasquez and Hester, 2013; Bangui et al., 2017; Whaiduzzaman et al., 2014) is a sub-area of operation research which is used in evaluating multiple conflicting criteria and choosing the optimal solution from several alternatives. Six criteria and four alternatives are considered in the proposed work. AHP is used to assign weights for each criterion and TOPSIS is used to choose ideal cloud.

4.1 Analytic hierarchy process

AHP is centred on pairwise comparison among different criteria arranged in hierarchical form. The highest level of the hierarchy is the goal and lower level is the criteria and the alternative to be selected. The best alternative is chosen after evaluating the criteria. In the proposed work, the goal is to choose the best cloud server among a class of available cloud servers which are offering similar services. The six different types of criteria considered are bandwidth, speed, proximity, availability, security and cost. The alternatives are four cloud servers offering the same service located in different regions. One of the cloud servers, located in Asia Mumbai region is a higher end server compared to the other three servers located in Asia Mumbai, Asia Singapore and North California. Figure 4 represents the decision hierarchy based on the criteria and alternatives.

Step 1 Pairwise comparison matrix is represented as $A[n \times n]$ where n is the number of evaluation criteria. Each entry a_{ij} indicates the importance of i^{th} criteria relative to j^{th} criteria as per the scale of relative importance as given in Table 2 (Wu et al., 2012). For example, if $a_{ij} > 1$, it indicates that i^{th} criterion is more important than the j^{th} criterion. Table 2 is used to translate the decision makers qualitative evaluation into quantitative. Intermediate values can be assigned as given in Table 2.

Step 2 Normalised pairwise matrix A_{norm} is formulated where each element in pairwise matrix A is divided by the column wise sum of each criterion of pairwise comparison matrix A.

Step 3 Criteria weight vector w_j is obtained by finding the sum of each row of the normalised pairwise matrix A_{norm} and dividing the sum by number of criterions.

Step 4 Steps 5 to 8 is done to check if the consistency of calculated criteria weight is correct.

Step 5 Weighted sum value of each criterion is calculated by finding row sum of each criteria.

Step 6 λ_{max} value is calculated by finding the sum of ratio of weighted sum value and criteria weights divided by the criteria count.

Step 7 Consistency index (CI) is obtained by the formula given in equation (1) where the number of criteria is represented by n.

$$CI = \frac{\lambda_{max} - n}{n - 1} \tag{1}$$

Step 8 Consistency ratio is computed by CI/RI where RI is the random index taken from Table 3 based on number of criterions. The value considered is 1.24.

If consistency ratio is less than 0.10, it is assumed that the calculated metrics is reasonably consistent.

Figure 4 Decision hierarchy

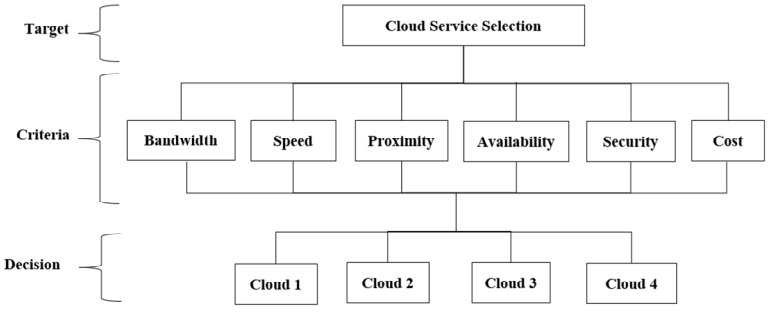


Table 2 Scale of importance

Definition	*Intense of importance*
Equally important	1
Moderately important	3
Strongly important	5
Very strongly important	7
Extreme important	9
Intermediate comparison	2, 4, 6, 8

Table 3 Random index table

n	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Random index	0	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

4.2 TOPSIS

This technique selects an approach which is close to ideal solution and far from anti-ideal solution. Alternatives and criteria are represented in a matrix form of $a[m \times n]$ where number of alternatives is represented by m and number of criteria is represented by n.

Step 1 Normalised matrix is obtained using equation (2) where the row of matrix contains the alternatives and column of the matrix the criteria.

$$\bar{X}_{ij} = \frac{X_{ij}}{\sqrt{\sum_{i=1}^{n} X_{ij}^2}} \tag{2}$$

Step 2 Weighted normalised matrix is calculated using equation (3) where weights for each criterion is multiplied with each element of normalised matrix.

$$V_{ij} = \bar{X}_{ij} \times W_j \tag{3}$$

Step 3 The ideal best and worst for each criterion is calculated. If the criteria are beneficial then the ideal best is the max value of the criteria and the min. value of the criteria is the ideal worst. When the criteria are non-beneficial, then the ideal best is the min value of the criteria and max. value of the criteria is ideal worst.

Step 4 Equation (4) and equation (5) is used to find the Euclidean distance from the ideal best and ideal worst where V_j^+ is ideal best and V_j^- is the ideal worst of each alternative.

$$S_i^+ = \left[\sum_{j=1}^{m} \left(V_{ij} - V_j^+ \right)^2 \right]^{0.5} \tag{4}$$

$$S_i^- = \left[\sum_{j=1}^{m} \left(V_{ij} - V_j^- \right)^2 \right]^{0.5} \tag{5}$$

Step 5 Performance score is calculated for each alternative as given in equation (6).

$$P_i = \frac{S_i^-}{S_i^+ + S_j^-} \tag{6}$$

Step 6 The rank is allocated to each alternative based on the performance score. Rank 1 is assigned to the alternative with highest performance score, the next highest performance score is ranked 2 and so on in the descending order of the performance score.

4.3 Performance analysis of cloud service selection

Performance analysis of cloud service selection based on theoretical analysis is compared with experimental analysis done on actual cloud servers, considering four cloud servers located at different regions. The weights for each criterion are calculated using AHP. AHP helps the decision makers in assigning priorities between criterions to make better decisions. The weights of the criteria obtained using AHP after performing the steps discussed in Section 4.1 is as depicted in Table 4. The consistency of obtained weights is checked to minimise the bias during the decision-making process. The consistency ratio CR is 0.055 which is less than 0.1 which is the standard check point. Since CR 0.055 < 0.1, the weights obtained are reasonably consistent and is used in the decision-making process. After obtaining criterion weights, TOPSIS is used to rank the different alternatives. The alternatives are the different cloud servers.

Bandwidth is most important criteria because it involves communication cost between mobile device and cloud server or cloud server to mobile device. Speed is subsequently important because it improves the execution time of task. Hence, the importance of criteria is ranked in the order bandwidth > speed > proximity > availability > security > cost.

Table 4 Criteria weights obtained from AHP

Criteria	*Weights*	*Weighted sum value*	λ_{max}	*CI*	*RI*	*CR*
Bandwidth	0.34	2.31	6.339	0.0677	1.24	0.055
Speed	0.27	1.87				
Proximity	0.22	1.44				
Availability	0.1	0.6				
Security	0.04	0.24				
Cost	0.02	0.15				

It is observed that criteria weights can have a substantial influence on the result of the decision-making process. During the experiment, entropy and critic methods were tried for assigning weights to the criteria which are objective methods where mathematical functions are fully utilised to calculate the criteria weights. As results obtained were unpromising, AHP was chosen for assigning weights of the criteria. AHP is subjective method where decision maker's inputs are critical in assigning weights to the criteria.

4.3.1 Theoretical analysis

Based on six criteria and four alternatives, the optimal cloud is ranked using TOPSIS. Speed and cost are higher for cloud server 4 and equal for all other remaining cloud servers since cloud server 4 has higher processing power compared to the other cloud servers. Considering the user location to be Bangalore, Mumbai is closer followed by Singapore and California. Hence, the proximity criteria were set lower for Mumbai followed by Singapore and California. Other criterions were given the same importance for all the alternatives. Table 5 represents the results obtained using the above assumptions and TOPSIS algorithm.

Table 5 Ranking of cloud server using theoretical analysis

Alternatives			*Si+*	*Si–*	*Pi*	*Rank*
Server	*Location*	*Size*				
Cloud 1	California	T2.Medium	0.1523	0.011	0.0671	4
Cloud 2	Singapore	T2.Medium	0.1161	0.058	0.3332	3
Cloud 3	Mumbai	T2.Medium	0.1011	0.1144	0.5308	2
Cloud 4	Mumbai	C4.2xlarge	0.011	0.1523	0.9329	1

4.3.2 Experimental analysis

The real time values obtained based on the execution of the application is considered for the criteria. The application considered is to solve the mathematical puzzle Tower of Hanoi. The number of the disks considered for the experiment was in the range of 12 to 22 disks. The computation time, communication time and total time taken for execution was calculated for the number of disks ranging from 12 to 22 disks. The experiment was performed ten times for each disk and average of each instance is taken to maintain consistency of the reading. The experiment was repeated on four different cloud servers.

Table 6 Criterion values assigned for different alternatives

Alternatives			*Criteria*			
Server	*Location*	*Size*	*Bandwidth/communication cost*	*Speed/computation cost*	*Proximity/distance*	*Cost*
Cloud 1	California	T2.medium	0.94388	0.4144	14194	0.0552
Cloud 2	Singapore	T2.medium	0.22108	0.4144	7050.5	0.0584
Cloud 3	Mumbai	T2.medium	0.1842	0.4144	981.2	0.0496
Cloud 4	Mumbai	C4.2xlarge	0.1764	0.3134	981.2	0.4

The above experimental results were considered as the criteria values for the selection of optimal cloud using TOPSIS. Average communication time taken for executing the entire experiment on different cloud servers were given as input to the bandwidth criteria for different alternative. Average computation time taken for executing the entire experiment on different cloud servers were given as input to the speed criteria for different alternative. The cost of each cloud server per hour was given as input to the cost criteria. The distance from the current location to each cloud server was calculated and given as input to the proximity criteria. The criteria value for different alternatives is as show in Table 6.

The other two criteria availability and security were given equal importance for all the alternatives. The results obtained using TOPSIS to rank the optimal cloud is as represented in Table 7.

Table 7 Ranking of cloud server using experimental analysis

Alternatives			*Si+*	*Si–*	*Pi*	*Rank*
Server	*Location*	*Size*				
Cloud 1	California	T2.Medium	0.323	0.0211	0.0614	4
Cloud 2	Singapore	T2.Medium	0.0922	0.268	0.7441	3
Cloud 3	Mumbai	T2.Medium	0.0346	0.3196	0.9023	2
Cloud 4	Mumbai	C4.2xlarge	0.0215	0.3229	0.9376	1

It is observed that the ranking of the servers remains the same with slight difference in the performance index score by comparing results obtained in Table 5 and Table 7. In theoretical analysis, values were assumed and in experimental analysis, the actual values obtained from series of experiments conducted are taken as input for the criteria values. The results prove that AHP and TOPSIS are viable solution for cloud path selection.

4.4 Time and power measurement

The execution delay in offloading the task onto the cloud server is taken as sum of communication time from mobile device to cloud server, execution time on the cloud server and communication time from cloud server to mobile device. The time taken for execution on cloud server should be lesser than time taken on mobile device for performance improvement of the mobile device.

Time taken to execute on mobile device (De et al., 2020) is computed as given in equation (7)

$$T_{Loc} = I / S_m \tag{7}$$

T_{Loc} execution time on mobile device

I count of instructions in the task to be executed

S_m speed of the mobile device.

Execution time on cloud server is computed as given in equation (8),

$$T_{Server} = T_{sdelay} + I / S_{cloud} + T_{rdelay} \tag{8}$$

T_{Server} execution time on cloud server

T_{sdelay} delay in transmitting request from mobile device to cloud server

I count of instructions in the task to be executed

S_{cloud} speed of the cloud server

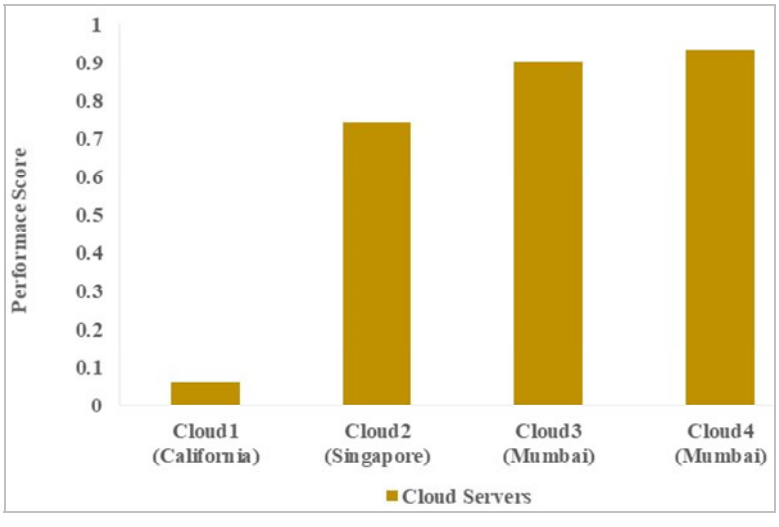
T_{rdelay} delay in receiving the response from cloud server to mobile device.

PowerTutor (2009) is utilised to analyse the energy consumption of mobile device when the job is executed on mobile device and cloud server.

5 Results and discussion

Figure 5 depicts the performance score of each cloud server. The cloud server with higher performance score is the optimal cloud server compared to other cloud servers. Cloud 4 located in Mumbai region with higher processing power C4.2xlarge is considered the optimal cloud server, followed by cloud 3 located in Mumbai region T2.Medium size, cloud 2 located in Singapore T2.Medium size and cloud 1 located in California T2.Medium size. The experimental results indicate that proximity of the server is an important criterion, the closer the server from the end user the faster the response.

Figure 5 Performance score of cloud servers (see online version for colours)



Execution of computationally intensive task on cloud server is better than execution on mobile device to enhance performance of mobile device (Sindhu and Guruprasad, 2020). The following section discusses the comparison results obtained by executing the task on the mobile device, offloading the task on the cloud server and fetching data from the cache. The Tower of Hanoi problem was considered by varying the number of disks from 12 to 22. The delay in executing the task and mobile device energy consumption were considered for the following three cases:

- fetching data from the cache

- execution of task on cloud server
- execution of task on mobile device.

5.1 Case 1: fetching data from the cache

On availability of data in the cache, data is fetched from the cache and displayed to the user. The mobile device energy consumption and execution time in fetching the data from the cache is calculated. The delay in fetching the data from the cache is between 0.005 to 0.0081 seconds. The cache was varied from 10 to 40 rows of data. Mobile device energy consumption when fetching the data from the cache is between 0.132 to 0.171 joules. It is observed that cache hit results in faster execution and low energy consumption of the mobile device. The experiment was reiterated ten times for each number of disks for all the three cases and average reading was considered for latency of execution and energy consumption.

5.2 Case 2: execution of task on cloud

On unavailability of data in the cache, the network availability of the mobile device is checked. If the communication network is available, the decision manager chooses the optimal cloud server and then a request for execution of the job is made to the cloud server. Requested module is executed and response is delivered back to mobile device. The fetched result is saved in the cache.

Figure 6 Time analysis of execution on different cloud servers (see online version for colours)

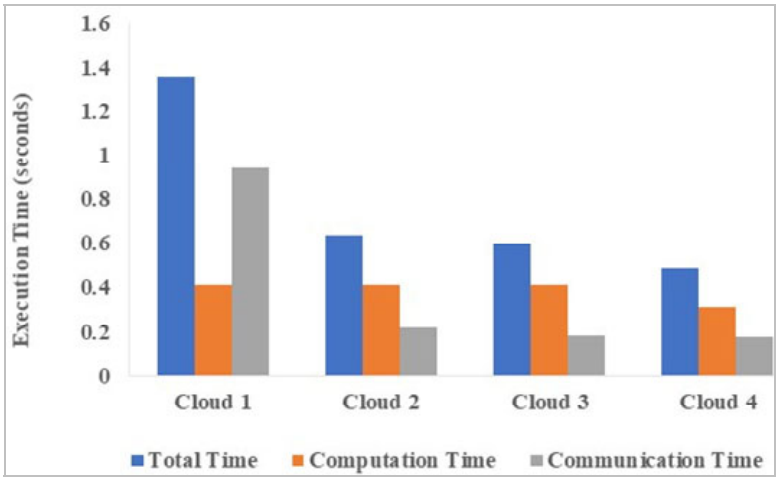


Figure 6 represents the computation time, communication time and total execution time taken in executing the task on different cloud servers. Specifications of each cloud server are represented in Table 1. Computation time taken on cloud server 1, cloud server 2 and cloud server 3 are the same since the infrastructure are same for all the three cloud servers. Cloud server 4 has higher computational power compared to the other three cloud servers and hence is faster. The communication time indicates that the location of the server from the current user location is an important criterion to be considered when

choosing the optimal cloud. Hence in the proposed work, proximity was considered as a criterion to decide the best alternative. Cloud server 1 is in North California and it takes longer time compared to the cloud server 2 in Singapore and cloud server 3 and cloud server 4 located in Mumbai region. Comparing the results obtained in Figure 6 and Table 7, we can conclude that AHP and TOPSIS are the viable solutions to find the optimal cloud server and cloud server 4 is the optimal cloud server.

5.3 Case 3: execution of task on mobile device

On unavailability of data in the cache and communication network, the execution of the job is done on mobile device. Result is displayed and the cache is updated.

Figure 7 Delay in execution on mobile device (see online version for colours)

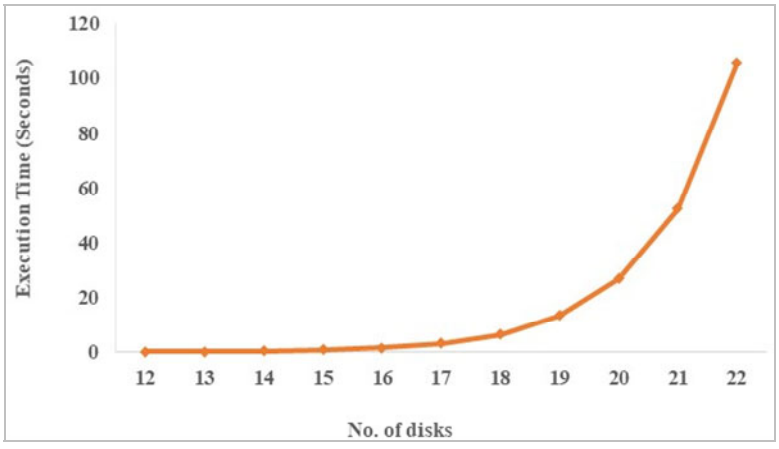
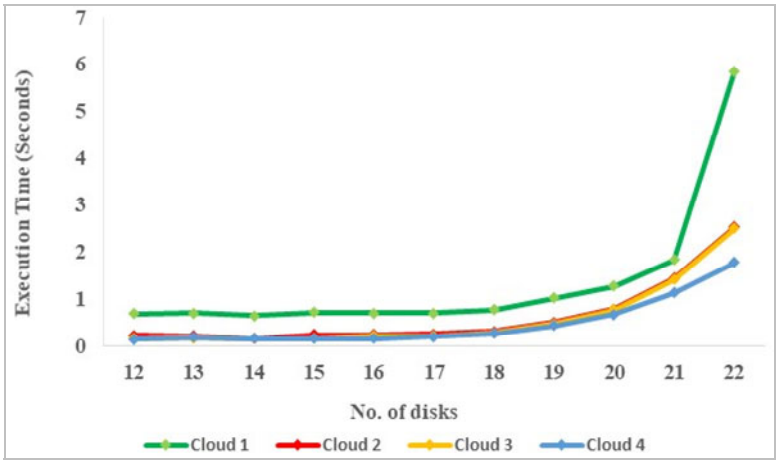


Figure 8 Delay in execution on different cloud servers (see online version for colours)



The comparison of delay in execution on mobile device and cloud servers are represented in Figure 7 and Figure 8, respectively. From the graph, it is evident that the execution at the cloud server is better as the number of disks increase. Cloud server 4 is faster

compared to the other cloud servers. When the number of the disks is 22, the latency of execution when executed on mobile device is 105.68 seconds, on cloud server 1 it is 5.85 seconds, cloud server 2 2.52 seconds, cloud server 3 2.50 seconds and cloud server 4 1.78 seconds. When a cache hit occurs, execution time was between 0.005 to 0.0081 seconds. The advantage of using the caching is to augment the performance of mobile devices by ensuring frequently used information is available and reducing the network calls. The framework also provides fault tolerance by sending the request to another cloud server in case response is not received from the requested cloud server after the timeout.

Figure 9 Energy consumption of mobile device during execution on mobile device (see online version for colours)

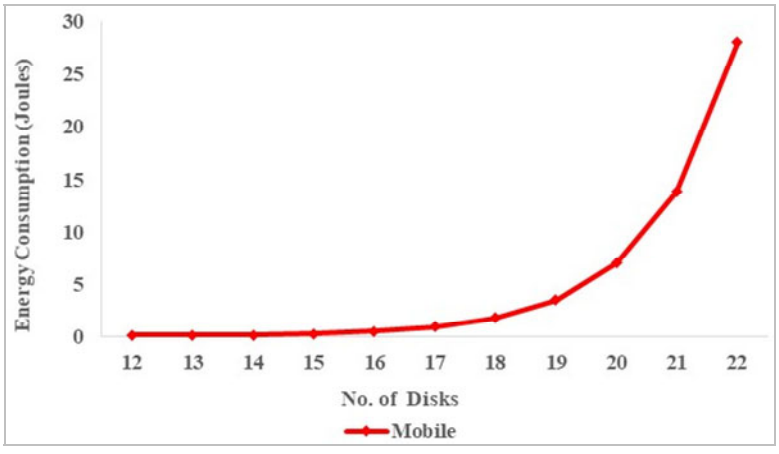
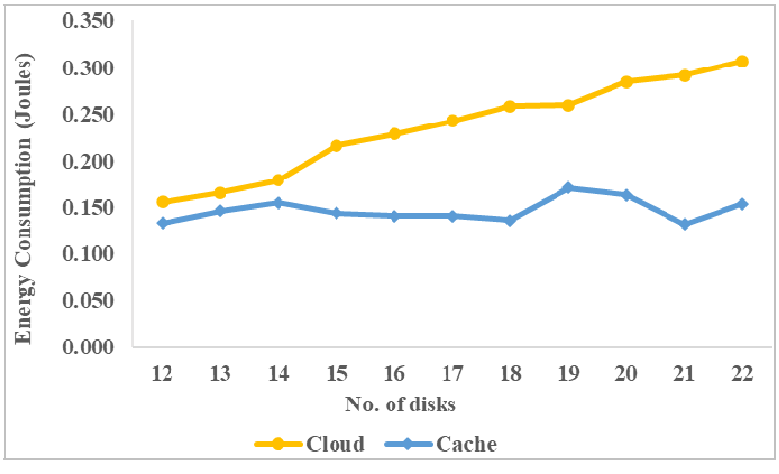


Figure 10 Energy consumption of mobile device during execution on server and cache (see online version for colours)



The mobile device energy consumption is reduced when the job is offloaded to cloud, or the data is fetched from the cache as represented in Figure 10 compared to executing on mobile device as given in Figure 9. When the number of the disks is 22, the energy

consumption of mobile device during execution on mobile device is 28.04 joules and cloud server it is 0.307 joules. When a cache hit occurs, energy consumption was between 0.132 to 0.171 joules. The main aim of the proposed work was to reduce application execution latency and energy consumption of mobile device which is achieved by finding optimal cloud server for offloading and incorporating a caching mechanism for the application. Based on the experiment conducted, the results indicate a performance increase of 99% in execution time and 97% in energy consumption using the caching scheme and 92% in execution time and 90% in energy consumption using offloading on cloud server compared to executing on the mobile device. Caching at edge servers can be considered for future work since the cache considered was of minimal size with existing storage constraints of mobile devices.

6 Conclusions

In the proposed work, the resource intensive task of an application is offloaded to an optimal cloud server using MCDA algorithm considering four cloud servers located at different regions. The experimental results are compared with theoretical analysis which proves that AHP and TOPSIS are the viable solutions to find the optimal cloud server. To further improve efficiency of the framework, caching scheme was implemented to minimise the battery consumption and execution latency of the mobile device. Results indicate that the proposed strategy can be used for applications seeking faster execution and reduced battery consumption.

References

Akherfi, K., Gerndt, M. and Harroud, H. (2018) 'Mobile cloud computing for computation offloading: issues and challenges', *Applied Computing and Informatics*, Vol. 14, No. 1, pp.1–16 [online] https://doi.org/10.1016/j.aci.2016.11.002.

Al-Janabi, S., Al-Shourbaji, I., Shojafar, M. and Abdelhag, M. (2017) 'Mobile cloud computing: challenges and future research directions', in *2017 10th International Conference on Developments in Esystems Engineering (DeSE)*, IEEE, June, pp.62–67, DOI: 10.1109/DeSE.2017.21.

Alsubhi, K., Imtiaz, Z., Raana, A., Ashraf, M.U. and Hayat, B. (2020) 'MEACC: an energy-efficient framework for smart devices using cloud computing systems', *Frontiers of Information Technology & Electronic Engineering*, Vol. 21, No. 6, pp.917–930.

Android (2017) *Building an App with Offline Support* [online] https://proandroiddev.com/build-an-app-with-offline-support-1a32c6bab7d2 (accessed 15 January 2021).

Android (2019) *Implementing Caching in Android* [online] https://blog.mindorks.com/implement-caching-in-android-using-rxjava-operators (accessed 15 January 2021).

Android (2021) *Volley* [online] https://developer.android.com/training/volley (accessed 2 January 2021).

Arun, C. and Prabu, K. (2017) 'Applications of mobile cloud computing: a survey', in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, IEEE, June, pp.1037–1041, DOI: 10.1109/ICCONS.2017.8250623.

Bangui, H., Ge, M., Buhnova, B., Rakrak, S., Raghay, S. and Pitner, T. (2017) 'Multi-criteria decision analysis methods in the mobile cloud offloading paradigm', *Journal of Sensor and Actuator Networks*, Vol. 6, No. 4, p.25, DOI: 10.3390/jsan6040025.

De, D., Mukherjee, A. and Roy, D.G. (2020) 'Power and delay efficient multilevel offloading strategies for mobile cloud computing', *Wireless Personal Communications*, Vol. 112, No. 4, pp.2159–2186 [online] https://doi.org/10.1007/s11277-020-07144-1.

Dinh, H.T., Lee, C., Niyato, D. and Wang, P. (2013) 'A survey of mobile cloud computing: architecture, applications, and approaches', *Wireless Communications and Mobile Computing*, Vol. 13, No. 18, pp.1587–1611 [online] https://doi.org/10.1002/wcm.1203.

Dutta, K. and Vandermeer, D. (2017) 'Caching to reduce mobile app energy consumption', *ACM Transactions on the Web (TWEB)*, Vol. 12, No. 1, pp.1–30 [online] https://doi.org/10.1145/3125778.

Enzai, N.I.M. and Tang, M. (2014) 'A taxonomy of computation offloading in mobile cloud computing', in *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, IEEE, April, pp.19–28, DOI: 10.1109/MobileCloud.2014.16.

Fernando, N., Loke, S.W. and Rahayu, W. (2013) 'Mobile cloud computing: a survey', *Future Generation Computer Systems*, Vol. 29, No. 1, pp.84–106 [online] http://dx.doi.org/10.1016/j.future.2012.05.023.

Goudarzi, M., Zamani, M. and Haghighat, A.T. (2017) 'A genetic-based decision algorithm for multisite computation offloading in mobile cloud computing', *International Journal of Communication Systems*, Vol. 30, No. 10, p.e3241.

Jadad, H., Touzene, A., Day, K. and Alzeidir, N. (2018) 'A cloud-side decision offloading scheme for mobile cloud computing', *International Journal of Machine Learning and Computing*, Vol. 8, No. 4, pp.367–371, DOI: 10.18178/ijmlc.2018.8.4.713.

Liang, R., Zhong, Y. and Xia, Q. (2018) 'Energy-saved data transfer model for mobile devices in cloudlet computing environment', in *2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, IEEE, April, pp.271–274, DOI: 10.1109/ICCCBDA.2018.8386525.

Magurawalage, C.M.S., Yang, K., Hu, L. and Zhang, J. (2014) 'Energy-efficient and network-aware offloading algorithm for mobile cloud computing', *Computer Networks*, Vol. 74, pp.22–33 [online] https://doi.org/10.1016/j.comnet.2014.06.020.

Malavolta, I., Chinnappan, K., Jasmontas, L., Gupta, S. and Soltany, K.A.K. (2020) 'Evaluating the impact of caching on the energy consumption and performance of progressive web apps', in *Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems*, July, pp.109–119.

Noor, T.H., Zeadally, S., Alfazi, A. and Sheng, Q.Z. (2018) 'Mobile cloud computing: challenges and future research directions', *Journal of Network and Computer Applications*, Vol. 115, pp.70–85 [online] https://doi.org/10.1016/j.jnca.2018.04.018.

Odu, G.O. (2019) 'Weighting methods for multi-criteria decision making technique', *Journal of Applied Sciences and Environmental Management*, Vol. 23, No. 8, pp.1449–1457, DOI: 10.4314/jasem.v23i8.7.

Othman, M., Madani, S.A. and Khan, S.U. (2013) 'A survey of mobile cloud computing application models', *IEEE Communications Surveys & Tutorials*, Vol. 16, No. 1, pp.393–413, DOI: 10.1109/SURV.2013.062613.00160.

PowerTutor (2009) *A Power Monitor for Android-based Mobile Platforms* [online] http://ziyang.eecs.umich. edu/projects/powertutor/documentation.html (accessed 20 February 2014).

Safavat, S., Sapavath, N.N. and Rawat, D.B. (2020) 'Recent advances in mobile edge computing and content caching', *Digital Communications and Networks*, Vol. 6, No. 2, pp.189–194 [online] https://doi.org/10.1016/j.dcan.2019.08.004.

Saha, S. and Hasan, M.S. (2017) 'Effective task migration to reduce execution time in mobile cloud computing', in *2017 23rd International Conference on Automation and Computing (ICAC)*, IEEE, September, pp.1–5.

Sarvabhatla, M., Konda, S., Vorugunti, C.S. and Babu, M.N. (2017) 'A network aware energy efficient offloading algorithm for mobile cloud computing over 5G network', in *2017 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, IEEE, November, pp.69–74.

Sindhu, K. and Guruprasad, H.S. (2020) 'Mobile device performance enhancement using computational offloading', *International Journal of Advanced Science and Technology*, Vol. 29, No. 5, pp.5607–5617.

Singla, C. and Kaushal, S. (2015) 'Cloud path selection using fuzzy analytic hierarchy process for offloading in mobile cloud computing', in *2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS)*, IEEE, December, pp.1–5.

Somula, R.S. and Sasikala, R. (2018) 'A survey on mobile cloud computing: mobile computing + cloud computing (MCC = MC + CC)', *Scalable Computing: Practice and Experience*, Vol. 19, No. 4, pp.309–337.

Thanapal, P. and Durai, M.S. (2018) 'Energy saving offloading scheme for mobile cloud computing using CloudSim', *International Journal of Advanced Intelligence Paradigms*, Vol. 10, Nos. 1–2, pp.45–62.

Velasquez, M. and Hester, P.T. (2013) 'An analysis of multi-criteria decision making methods', *International Journal of Operations Research*, Vol. 10, No. 2, pp.56–66.

Whaiduzzaman, M., Gani, A. and Naveed, A. (2015) 'Towards enhancing resource scarce cloudlet performance in mobile cloud computing', *Computer Science & Information Technology*, p1, DOI: 10.5121/csit.2015.50401.

Whaiduzzaman, M., Gani, A., Anuar, N.B., Shiraz, M., Haque, M.N. and Haque, I.T. (2014) 'Cloud service selection using multicriteria decision analysis', *The Scientific World Journal*, Vol. 2014, p.10 [online] https://doi.org/10.1155/2014/459375.

Wu, H. (2018) 'Multi-objective decision-making for mobile cloud offloading: a survey', *IEEE Access*, Vol. 6, pp.3962–3976, DOI: 10.1109/ACCESS.2018.2791504.

Wu, H., Wang, Q. and Wolter, K. (2012) 'Methods of cloud-path selection for offloading in mobile cloud computing systems', in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, IEEE, December, pp.443–448.

Zhao, H., Chen, M., Qiu, M., Gai, K. and Liu, M. (2016) 'A novel pre-cache schema for high performance Android system', *Future Generation Computer Systems*, Vol. 56, pp.766–772 [online] https://doi.org/10.1016/j.future.2015.05.005.

Zhao, Y., Wat, P., Laser, M.S. and Medvidović, N. (2018) 'Empirically assessing opportunities for prefetching and caching in mobile apps', in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, September, pp.554–564 [online] https://doi.org/10.1145/3238147.3238215.

Zhou, B., Dastjerdi, A.V., Calheiros, R.N., Srirama, S.N. and Buyya, R. (2015) 'A context sensitive offloading scheme for mobile cloud computing service', in *2015 IEEE 8th International Conference on Cloud Computing*, IEEE, June, pp.869–876, DOI: 10.1109/CLOUD.2015.119.