DOI 10.12694/scpe.v20i4.1570
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 4, pp. 687–698. http://www.scpe.org 

# AN EFFECTIVE CONTEXT SENSITIVE OFFLOADING SYSTEM FOR MOBILE CLOUD ENVIRONMENTS USING SUPPORT VALUE-BASED CLASSIFICATION

MOSTAFA ABDULGHAFOOR MOHAMMED[*] AYA AHKAM KAMIL[†] RAED ABDULKAREEM HASAN[‡] AND NICOLAE ŢĂPUŞ[§]

**Abstract.** Mobile cloud computing (MCC) has drawn significant research attention recently due to the popularity and capability of portable devices. This paper presents an MCC offloading system based on internet offloading choices. This system guarantees the conservation of battery life and reduced execution time. The proposed effective context sensitive offloading approach using support value-based classification is processed in different steps. Initially, the context data of the input tasks is extracted through the energy consumption model, cost model, execution model, communication model and stored. Then, the support value-based classification approach classifies the tasks based on the context information. This classification creates the information about the tasks and finally, a decision is made at the right time to achieve better offloading. The result indicates the presented offloading framework can choose reasonable cloud assets depending on various contexts of the mobile devices and achieve significant performance enhancement.

**Key words:** Context data extraction, Support value measure, Classification, Generated knowledge, Offloading decision

**AMS subject classifications.** 68Q25, 68R10, 68U05

**1. Introduction.** Mobile cloud computing is a recent model that integrated cloud computing [29] . It enables the portable clients to use cloud services of their interest. It is imagined that this worldview will help to defeat the restrictions of the cell phones equipment [14, 32]. Offloading is an essential procedure in MCC [5]. It offloads the assignment to a cloud benefit by means of remote systems. Because of the generally low battery limit of cell phones (just as delicate versatile systems), several explorations have been performed on offloading to the cloud to accomplish two principle goal: to extend battery lifetime [5, 16, 7, 27], and to shorten the execution time of substantial uses [28, 9, 24, 13]. Offloading is a significant method in MCC [21, 23] that has been extensively investigated. For instance, in Think Air [16] , versatile side offloading gathers the runtime information and feed them into the energy estimation display. The energy estimation is also actualized inside the Think Air energy profiler on the versatile side. It is used to evaluate the utilization of each running strategy [4]. Like Think Air, many offloading frameworks [1, 8, 2] have executed the profilers, cost forecast models, and choice motor segments on the versatile side.

Numerous works have been done in MCC [17, 10, 19, 20] and most of them focused on the offloading strategies, accepting a steady system and adequate transfer speed. This sort of offloading approaches keeps the offloading choice parts running amid application execution to settle on ideal offloading choices at runtime. Be that as it may, this procedure of settling on an offloading choice delivers a noteworthy overhead on the cell phone. This overhead is expensive on cell phones in terms of battery utilization and handling assets [30, 15].

To process the issues stated above and improve the service performance in mobile cloud computing, we propose an effective context sensitive offloading approach using support value-based classification. The proposed offloading approach is a promising method to enhance the execution process and reduce battery utilization in versatile applications. The target of the proposed framework is to determine an ideal offloading choice under the setting of the cell phone and cloud assets to give a better execution and less battery utilization. The foremost contributions of this work are as follows:

- Various contexts information is extracted via distinctive models, such as the energy consumption model, communication model, cost model, and execution model.
- Support value is estimated for the separate context information of each task and this measure is utilized for the viable context sensitive offloading in MCC.

[*]Faculty of Automatic Control and Computers, University Polytechnic of Bucharest 313 Splaiul Independentei, 060042, Romania. (alqaisy86@gmail.com).

[†]Middle Technical University Electrical Engineering Technical College, Baghdad, Iraq. (aa_eng_89@yahoo.com).

[‡]Faculty of Al-dour Technical institute, Northern Technical University, Mosel, 41002, Iraq (raed.isc.sa@gmail.com).

[§]Faculty of Automatic Control and Computers, University Polytechnic of Bucharest 313 Splaiul Independentei, 060042, Romania (nicolae.tapus@cs.pub.ro).

- Analyse the total processing energy of each task between two configurations based on Support value evaluation.

J48 is an extension of ID3. The additional features of J48 are accounting for missing values, decision trees pruning, continuous attribute value ranges, derivation of rules, etc. In the WEKA data mining tool, J48 is an open source Java implementation of the C4.5 algorithms. Where JRIP It implements a propositional rule learner called as Repeated Incremental Pruning to Produce Error Reduction (RIPPER) and uses sequential covering algorithms for creating ordered rule lists. The algorithm goes through 4 stages: Growing a rule, Pruning, Optimization and Selection. IBK or K-Nearest Neighbor classification classifies instances based on their similarity [3]. It is a type of Lazy learning where the function is only approximated locally and all computation is deferred until classification. An object is classified by a majority of its neighbors. K is always a positive integer. The layout of this paper is as follows: Section 2 surveys the related works regarding the proposed strategy. In sections 3, a short discussion about the proposed system is given; section 4 examines the exploratory outcomes, and section 5 finalized the study.

**2. Related Work and System Overview.** In this section, an overview of the proposed framework and the literature works was presented. Numerous related works in offloading for MCC were considered, including offloading decision amongst cell phones and open cloud administrations. Cloud offloading is delicate to the numerous constraints of the framework based on the context of the gadget. Subsequently, current works have proposed new frameworks to address these difficulties in MCC and to empower intermittent observing of numerous advanced measurements used to gather data. Warley Junior et al. [15] proposed a Context-Sensitive Offloading System (CSOS) that exploits the primary machine-picking up thinking methods and strong profiling framework to give offloading choices and increase the level of precision. The method failed to meet expectations when offloading choices overlooks logical data. CSOS executes all assignments identified with the choice on the portable device since the executing demand tasks identified with the preparation and testing of the categorization techniques are accomplished in the arranged period of the framework as a discontinuous procedure. In this manner, the expense of handling at running time is short; meanwhile, the portable devices possibly need to parse the preparation method to choose when to offload information. Tianhui Meng et al. [26] introduced and assessed a protected and effective offloading plan that was a mixture of irregular fillings. To proceed to a quantifiable dealing, a crossbreed Continuous time Markov chain (CTMC) was established; They proposed the security measurements which framework modelers need to settle on educated exchange off choices, including framework security.

Yongin Kwon et al. [18] displayed a specific execution offloading for applications with dynamic conduct in MCC. Execution offloading which moves a string between a cell phone and a server was regularly utilized. In such execution offloading procedures, it was run off the mill to progressively choose the code part to be offloaded through basic leadership calculations. To accomplish an ideal offloading execution, nonetheless, the gain and cost of offloading must be anticipated precisely for such methods. Fangxiaoqi Yu et al. [31] proposed an innovative dynamic mobility aware partial offloading (DMPO) technique to make sense of the measure of information progressively, composed with choice of correspondence way, limiting the energy utilization while fulfilling defer requirement. The suggested technique calculates the opportunity to subsequent transfer and allocates information size to all vacancy in that stage to accomplish the set objectives. Xing Chen et al. [6] proposed a structure that supports versatile applications through the context-aware computation offloading ability. It is a best method to enhance the execution process and reduce the battery utilization. Table 2.1 presents the qualitative examination of the existing context-aware offloading techniques.

Table 2.1 showed a subjective assessment of the prevailing context-aware offloading methods. It presented the strategies used in making the offloading choice and determine when offloading will enhance execution. Moreover, it demonstrates whether the investigation assesses the execution of the technique to identify the level of right choices made. Contrary to the related works, there is still a gap between rapid growth of power consumption and power capacities of current mobile devices. The proposed offloading approach is a promising method to enhance the execution process and reduce battery utilization in versatile applications. The target of the proposed framework is to determine an ideal offloading choice under the setting of the cell phone and cloud assets to give a better execution and less battery utilization. our framework considers the distinctive models for the extraction of context information, such as execution model, communication model, cost model, and

TABLE 2.1
*Qualitative analysis of the existing context-aware offloading techniques*

| Framework | Decision support | Solutions | Main contribution |
|---|---|---|---|
| CSOS [33] | JRIP and J48 classifiers | High accuracy and context-sensitive. | Offloading decisions that achieved gains and energy efficacy |
| CTMC [22] | CTMC and queuing method | Secure and cost-efficient Offloading Policy | Cost-efficient offloading scheme |
| OMMC [11] | TOPSIS and Energy model | Managed the trade-off between time and energy consumption. | Context-aware |
| Context-aware computation offloading [25] | Advanced structure for context-aware computation offloading | Rightly choose the suitable cloud assets and offload mobile codes on request | Reduced execution time and power consumption |
| DMPxcO [12] | DMPO algorithm | Decreased energy utilization and achieved the delay limit | Better performance in meeting the delay constraint |
| f_Mantis [33] | Sparse Polynomial Regression | Better accuracy in offloading | Precise execution offloading |
| mCloud [22] | TOPSIS and Cost model | mCloud | Context-aware |
| CoSMOS [22] | Cost functions | CoSMOS | Context-aware |
| Energy effective offloading choice algorithm [11] | Lyapunov optimization algorithm | Minimized average energy consumption, response time, less computational complexity | Energy-efficient |

energy consumption model. Our proposed extricates the context information of input tasks via these models. Afterward, the support consideration for the context information of each assignment is estimated and tasks were arranged based on the settings information. This support consideration achieved better offloading. The outcomes demonstrate that the proposed technique outperformed the existing techniques in terms of accuracy, sensitivity, specificity, FPR, FNR, energy utilization, throughput, and execution time.

**3. Effective Context Sensitive Offloading Using Support Value Based Classification Approach.** This paper provides an effective context sensitive offloading framework for mobile cloud environment utilizing support value-based classification in several steps; at first, the input tasks context information is extracted through the distinctive models (energy consumption model, cost model, execution model, and communication models). At this point, the context information is stored as training data through the processing module. After that, the support value is estimated for the context information of each task and creates knowledge about the tasks before making the final decision in the right way to prompt achieving better offloading intensive tasks. The flow structure of the presented technique is shown in Figure 3.1.

**3.1. Tasks collection.** Firstly, the users present their tasks in the mobile cloud framework through user requests. The number of tasks gathered depend on the number of clients at the remote site, determined as:

$$T_k = t_1, t_2, t_3, ...t_n \tag{3.1}$$

where $T_k$ is the set of requests and every request contains distinctive assignments. The task demands are accumulated from the clients for each specific time and after that, the accompanying parameters are extracted from the tasks to gather the information of the assignments.

**3.2. Context data extraction of tasks.** In this stage, the developer characterized the components that are necessary to be considered as important data for the offloading choice. Here, we distinguished the elements such as the execution time, application nature, execution cost, data size, communication models, energy utilization by CPU, I/O operation, memory, and cache. Every single measurement can vary individually of the others notwithstanding some that occasionally vary after some time whereas others are static. Consequently,

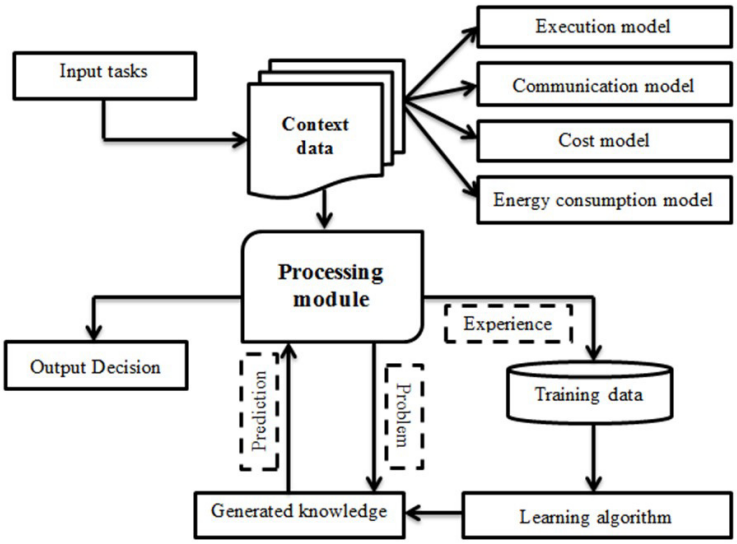


FIG. 3.1. *The flow structure of the presented decision module*

these different groupings can lead to the creation of various choices depending on the relevant data. Here, the context data of the input tasks separated by using the energy consumption model, cost model, execution model, and communication model are presented in Figure 3.1.

The decision are made according context data which is collected in terms of $\hat{E}_{con}$, $Ct_{mod}$, $\hat{C}_{mod}$ and $E_{time}$ where $\hat{E}_{con}$ Energy Consumption Model, $Ct_{mod}$ Cost model, $\hat{C}_{mod}$ Communication Model and $E_{time}$ Executions Model and these terms are explained in equitions to implement the methodology in processing model by using data traing to generate the knowledge and out the decision.

### 3.2.1. Energy Consumption Model ($\hat{E}_{con}$).

▷ **Energy consumption by CPU**

The energy consumed by the CPU is designated as $\tilde{E}_{cpu}$ and is calculated using Equation 3.2.

$$\tilde{E}_{cpu} = P_{avg} \cdot T_{exe} \tag{3.2}$$

where $P_{avg}$ signifies the average power utilized by the CPU, $T_{exe}$ signifies the execution time.

▷ **Energy utilization by I/O operation**

The energy utilization through I/O operation by a movable device is calculated using Equation 3.3.

$$\hat{E}_{I/O} = (\hat{T}_{elaps} - \hat{T}_{cpu}) \cdot \hat{P}_{I/O} \tag{3.3}$$

where $\hat{T}_{elaps}$, and $\hat{T}_{cpu}$ represents the overall time and CPU time essential for I/O processes, $\hat{P}_{I/O}$ signifies the power utilization for I/O activity.

▷ **Energy consumption by the memory**

The mobile device consumes its energy to reading and writes memory access. This energy is calculated using Equation 3.4.

$$\hat{E}_{memory} = (E_{read} \cdot \overline{read}) + (E_{write} \cdot \overline{write}) \tag{3.4}$$

where $E_{read}$ and $E_{write}$ are the energy used to execute read and write activities, respectively. This activity signifies the number of reads and writes, correspondingly.

▷ **Energy consumption by the cache**

A mobile device consumes energy for cache read and write and this energy is calculated using Equation 3.5.

$$\tilde{E}_{cash} = E_{cash1} + E_{cashL1} + E_{cashL2} \tag{3.5}$$

where $E_{cash1}$ , $E_{cashL1}$ and $E_{cashL2}$ represent the energy consumption due to the Instruction cache, level 1, and level 2 cache, respectively. The total energy utilization is computed using Equation 3.6.

$$\tilde{E} = \tilde{E}_{cpu} + \tilde{E}_{I\backslash O} + \tilde{E}_{memory} + \tilde{E}_{cash} \tag{3.6}$$

The residual energy of every mobile device can be determined using Equation 3.7.

$$\tilde{E}_{res} = \tilde{E}_T - \tilde{E}_{initial} \tag{3.7}$$

where $\tilde{E}_T$ and $\tilde{E}_{initial}$ represent the total and initial energy of a mobile device.

**3.2.2. Communication model ($\hat{C}_{mod}$).** The link between a mobile device and multi-sites is viewed as homogeneous; this suggests that the entire information exchange between the mobile device and any site is communicated at a similar rate. If is the power of a dynamic link, then, the energy utilization amid the transfer of the information between links is determined according to Equation 3.8:

$$\tilde{E}D_{active} = \hat{P}_{link-active} \cdot \sum_{i=1}^{t} DS(i,S) \tag{3.8}$$

where $DS(i,S)$ is the data size of module $i$ of an application offloaded on to site $(S)$.

Likewise, the energy expended during the information exchange amongst the sites is determined using Equation 3.9,

$$\tilde{E}S_{acive} = \tilde{P}_{site-active} \cdot \sum_{i=1}^{t} DS(i,S,k) \tag{3.9}$$

where $DS(i,S,k)$ is the task size of the module $i$ of an application offloaded on site $S$ from the site $k$ . $\tilde{P}_{site-active}$ is the power among the dynamic sites. The overall energy utilization throughout the transfer of information is the summation of Equations 3.8 and 3.9 as displayed in Equation 3.10.

$$\tilde{E}_{active} = \tilde{E}D_{active} + \tilde{E}S_{active} \tag{3.10}$$

**3.2.3. Execution model($E_{time}$).**

▷ **Execution time**

The execution time of the module $i$ on location $S$ is determined using Equation 3.11, i.e., the addition of the processing time and information transfer time.

$$E_{time}(i,S) = \frac{tz_i}{CP} + \frac{DS(i,S)}{B(i,S)} \tag{3.11}$$

where $CP$ is the cyclic prefix, $E_{time}(i,S)$ is the complete execution time of the entirely offloaded $z$ modules on the cloud, and is determined using Equation 3.12:

$$TE_{time}(i,S) = \sum_{i=1}^{z} E_{time}(i,S) \tag{3.12}$$

where $E_{time}(i,S)$ is the execution time and $TE_{time}(i,S)$ is the overall execution time of the model.

**3.2.4. Cost model ($Ct_{mod}$).**

▷ **Data size**

The data of an input task is signified in a sequence of bits. The size of the information is estimated in bits. Data is a set of values of subjects regarding qualitative or quantitative factors. Data and information are frequently used reciprocally; anyway, data move towards information when it is seen in context. Here, the size of the data is estimated as a context.

▷ **Execution cost**

The execution cost of the module $\hat{E}C(i,S)$ is the product of the unit charger rate and the execution time. This is expressed in Equation 3.13:

$$\hat{E}C(i,S) = \beta \times \hat{E}T(i,S) \tag{3.13}$$

where $\beta$ is the charge unit of utilizing the site for every time period; the total execution cost is determined using Equation 3.14:

$$\hat{E}C_T = \sum_{i=1}^{t} \hat{E}C(i,S) \tag{3.14}$$

The execution of a module on the quickest site is determined using Equation 3.14. These models might be utilized to enhance the evaluation of decision outcomes.

**3.3. Processing Module.** The component processing module gets the context information and form of the task, formulates a problem by describing the input data with the context data, and determine the support value for the context information of every task. The information about the context and choice is generated by characterizing the offloading decision which is done by classifying the context data using the support value-based classification method. The processing module chooses the information to be stored in the training database. The processing component is responsible for activities like processing of input information, storing the context data as a training data, classifying the gained information using the support value-based classification method, and executing the offloading procedure.

**3.3.1. Storing context data in the database.** Storing context data in the database refers to the loading of a training dataset with the context data. Here, all the extracted parameters from the execution model, communication model, energy consumption model, and cost model are stored as training context information. Toward the end of this stage, the support value is estimated for each context data of each task and stored for classification.

**3.3.2. Support value measure.** In this section, the context data of each task is classified based on the support value of the contexts. Here, the support value calculation depends on the extraction models such as execution model, energy consumption model, communication model, and the cost model. It is calculated using Equation 3.15.

$$\tilde{S}_{value} = (\hat{E}_{con} + \hat{C}_{mod} + Ct_{mod}) \;/\; (\hat{E}_{con} \cdot \hat{C}_{mod} \cdot Ct_{mod}) \tag{3.15}$$

where $\hat{E}_{con}$ is the energy utilization models' data, $\hat{C}_{mod}$ is the communication models' data, and $Ct_{mod}$ is the communication models' data.

**3.3.3. Offloading decision using support value-based classification.** In this section, the final offloading decision is made using the support value-based classification. Moreover, the information gained from the classified models should be saved in CSOS to consent the decision to load them at runtime. The pseudo code of the offloading decision model using support value-based classification is presented in Algorithm 1.

The pseudocode of the offloading decision-making given in Algorithm 1 deals with the context information and categorizes the utmost current task in the dataset using support value-based classification. Firstly, it gets the extricated context data and the support value-based classifier individually. The context information is extricated from the execution model, communication model, energy utilization model, and cost model. In any case, the estimation of this measurement is well-known at the time that the application's client chooses the

**Algorithm 1** Proceduer for affloading decision with classifier

```
begin (Context data, support value based classifier Packet Reception);
for each Contexts A ∈ to Database do
  Contexts[]contexts← get Contexts (A);
  for i = 1, tasks do
    if context.Get Name[i] = Data Size then
      Contexts[]contexts← get Contexts (A);
    else
      Instant.Set Value[i] ← Values[i];
    end if
  end for
end for
result ← Clasify Instant(support value based classfier, Instant);
if threshold(T_k) ⩽ result then
  response ← true;
  return response ;
else
  return response ;
end if
```

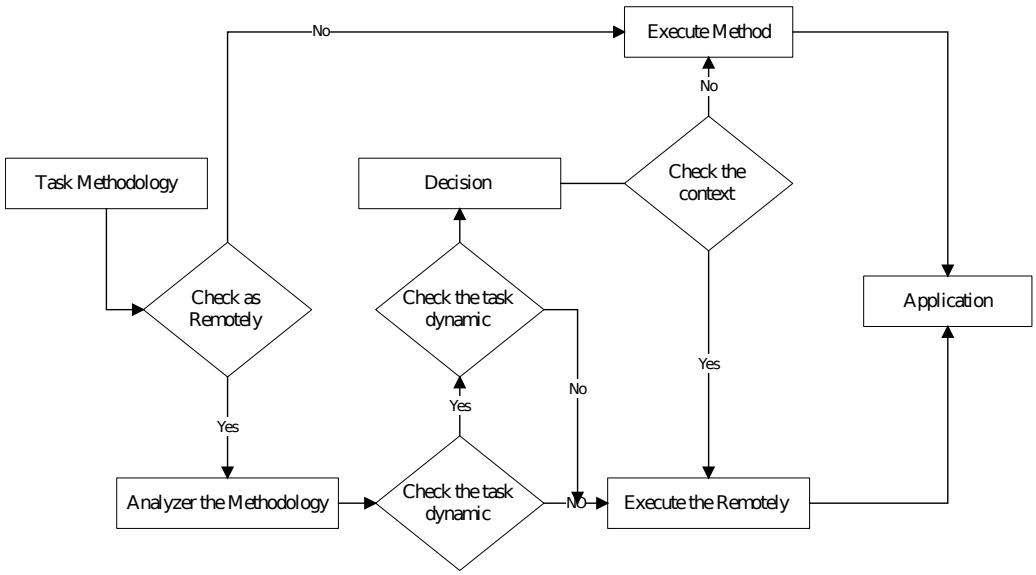


FIG. 3.2. *Execution flow diagram of the proposed offloading decision*

preferred context for processing. Subsequently, the profiling framework runs a search for a specific time which is unrealistic to precisely catch the estimation of this measurement and feed it to the decision engine. Next, every context of the utmost latest record in the database for data gathering is checked. At last, decision is made at the right time to accomplish more prominent advantages in offloading the computation-intensive tasks. The proposed offloading decision execution flow is illustrated in Figure 3.2.

Offloading decision explained in Figure 3.2 that show it gets the extricated context data and the support value-based classifier individually. In any case, the estimation of this measurement is well-known at the time that the application's client chooses the preferred context for processing. Subsequently, it excute the remolty with syncornization its excute the method the find the right decision for such appilaction

**4. Performance Measures.** The performance of the proposed work was examined using different parameters, such as accuracy, sensitivity, specificity, FPR, FNR, precision, energy consumption, CPU usage, throughput, and execution time as described in the following subsections.

**4.1. Accuracy.** Accuracy refers to the determination of the accuracy of an outcome in a populace (either positive or negative). It quantifies the level of data classification correctness. Accuracy is calculated using Equation 4.1:

$$\frac{(TN+TP)}{(TN+TP+FN+FP)} \tag{4.1}$$

where $TN$ and $TP$ are true negative and positive correspondingly, $FN$ and $FP$ are false negative and positive, correspondingly.

**4.2. Sensitivity.** Sensitivity is the measure of the effective differences between the in a classification task. It demonstrates the greatness of a system in data classification and can be calculated using Equation 4.2:

$$Sensitivity = \frac{(TP)}{(TP+FN)} \tag{4.2}$$

**4.3. Specificity.** Specificity refers to the measure of the effective differences between the by a classification method. It measures the ability of a method to distinguish between normal and abnormal information. Specificity can be calculated using Equation 4.3:

$$Specificity = \frac{(TN)}{(TN+FP)} \tag{4.3}$$

**4.3.1. FPR.** False positive rate (FPR) ascertains the extent negative events are incorrectly considered as positives, as well as the aggregate quantity of actual negative events. FPR is computed using Equation 4.4:

$$FPR = \frac{(FP)}{(FP+TN)} \tag{4.4}$$

**4.3.2. FNR.** False negative rate (FNR) is the number of positives which gives negative results. FNR is computed using Equation 4.5.

$$FPR = \frac{(TP)}{(TP+FP)} \tag{4.5}$$

**4.3.3. Precision.** Precision is the likelihood that information classification with a correct test result truly has precise information. Precision is commonly computed using Equation 4.6:

$$Precision = \frac{(TP)}{(TP+FP)} \tag{4.6}$$

**4.3.4. Throughput.** Throughput measures process performance per unit time. It is estimated by the fraction of the number of processes finished for a particular time. This is computed using Equation 4.7:

$$\widehat{T}_k = \overline{N}_f \ / \ t_p \tag{4.7}$$

where $\overline{N}_f$ is the amount of process finished, $t_p$ is the chosen time period, and $\widehat{T}_k$ is the throughput.

**4.3.5. Energy Consumption.** Energy consumption involves all the energy consumed when performing a task. Energy is mainly sourced from electric power ($E_{pi}$); execution time ($E_{exei}$) is the time required to process the $i^{th}$ execution up to a number of repetitions. Energy consumption is represented as:

$$\overline{E}_{Ci} = E_{pi} \cdot E_{exei} \tag{4.8}$$

The mean energy is computed using Equation 4.9:

$$\overline{E}_{mean} = \frac{\sum_i^n \overline{E}_{Ci}}{\widehat{n}} \tag{4.9}$$

where $\overline{E}_{mean}$ signifies the mean energy utilization and $\widehat{n}$ signifies the number of reiterations.

TABLE 4.1
*Comparison of the proposed technique in terms of different performance measures*

| Method | Accuracy | Specificity | Sensitivity | Precision | False positive rate | False negative rate | F1 |
|---|---|---|---|---|---|---|---|
| Proposed | 97.05 | 91.99 | 95.89 | 95.12 | 6.07 | 1.09 | 97.32 |
| J48 | 94.30 | 93.35 | 95.97 | 94.68 | 4.41 | 4.91 | 94.34 |
| JRIP | 94.35 | 88.93 | 94.77 | 91.01 | 5.79 | 4.54 | 93.59 |
| IBK | 93.12 | 90.09 | 94.15 | 92.34 | 4.70 | 4.79 | 94.75 |

**4.3.6. Execution Time.** This is the amount of time utilized to accomplish a given task. The time used to execute a process is calculated using Equation 4.10:

$$\widehat{E}_t = S_d \,/\, \widehat{B}_r \tag{4.10}$$

where $\widehat{E}_t$ signifies the execution time, $S_d$ represents the size of data, $\widehat{B}_r$ represents the bit rate. The evaluation table of the proposed technique compared to the existing techniques in terms of different performance measures is presented in Table 4.1.

**5. Results and Discussion.** The presented effective context sensitive offloading approach using support value-based classification was implemented in a JAVA platform with Cloudsim. To examine the performance of the presented work, different measures were evaluated. The performance of the proposed method is presented in Table 4.1 and Figure 5.1.

The graphical representation of the analysis of the proposed work is represented in this section. Figure 5.1 compared the performances of the classification algorithms using different indicators like accuracy, sensitivity, specificity, FPR, FNR, precision, energy consumption, CPU usage, throughput, and execution time.

Sensitivity and precision are the two major performance metrics used in applications that placed a premium on the successful detection of a class of events over the other classes. Unfortunately, these two measures are opposed to each other due to the differences between them; for instance, if we want to extract more relevant records (aimed at increasing the rate of sensitivity), more irrelevant records will also be retrieved (decreases the precision rate). Figures 5.1.b and 5.1.e clearly showed that the proposed technique has F1 greater than JRIP, J48, and IBK with a rate of 97.32%. Figure 5.1.a depicted the comparative analysis in terms of accuracy. High accuracy is important in MCC due to the dynamic and adaptive nature of mobile systems. Such nature leads to inaccurate decisions that later results in high energy utilization and performance degradation. The results of this study showed that the accuracy of the rules generated by the support-based algorithm was slightly higher than those of J48, JRIP, and IBK based on the contextual dataset. The proposed algorithm achieved an accuracy of 97.05 while JRIP and J48 algorithms achieved 94.35% and 94.30%, respectively. IBK achieved the worst accuracy level of 93.12% of correctly classified records. From these results, the good classifiers are the proposed algorithm, JRIP, and J48 based on their performances over the context database.

Figure 5.1.f showed the throughput of our proposed effective context sensitive offloading approach using support value-based classification and its CPU usage. The comparison graph of the proposed approach with the existing J48 classifier, JRIP classifier, IBK classifiers in terms of energy consumption was presented in Figure 5.1.g which illustrated the proposed algorithm to give better classification results compared to the existing J48 classifier, JRIP classifier, and IBK classifier. The comparison graph of the proposed approach with the existing classifiers in terms of execution time was presented in Figure 5.1.h, where the proposed approach was shown to give better classification results compared to the existing classifiers.

**6. Conclusion.** In this paper, we presented an effective context sensitive offloading approach using support value-based classification for context-aware computation offloading. Our framework utilizes diverse models to extricate the context data to ensure a progressive selection of suitable cloud assets and offload mobile codes to them on request. Additionally, the support value-based classification approach effectively generates information about the tasks before making a decision on the right time to accomplish a better offloading. The experimental results showed that our proposed approach outperformed the existing J48 classifier, JRIP classifier, and IBK classifier in terms of accuracy, sensitivity, specificity, FPR, FNR, precision, throughput, energy utilization, and execution time.

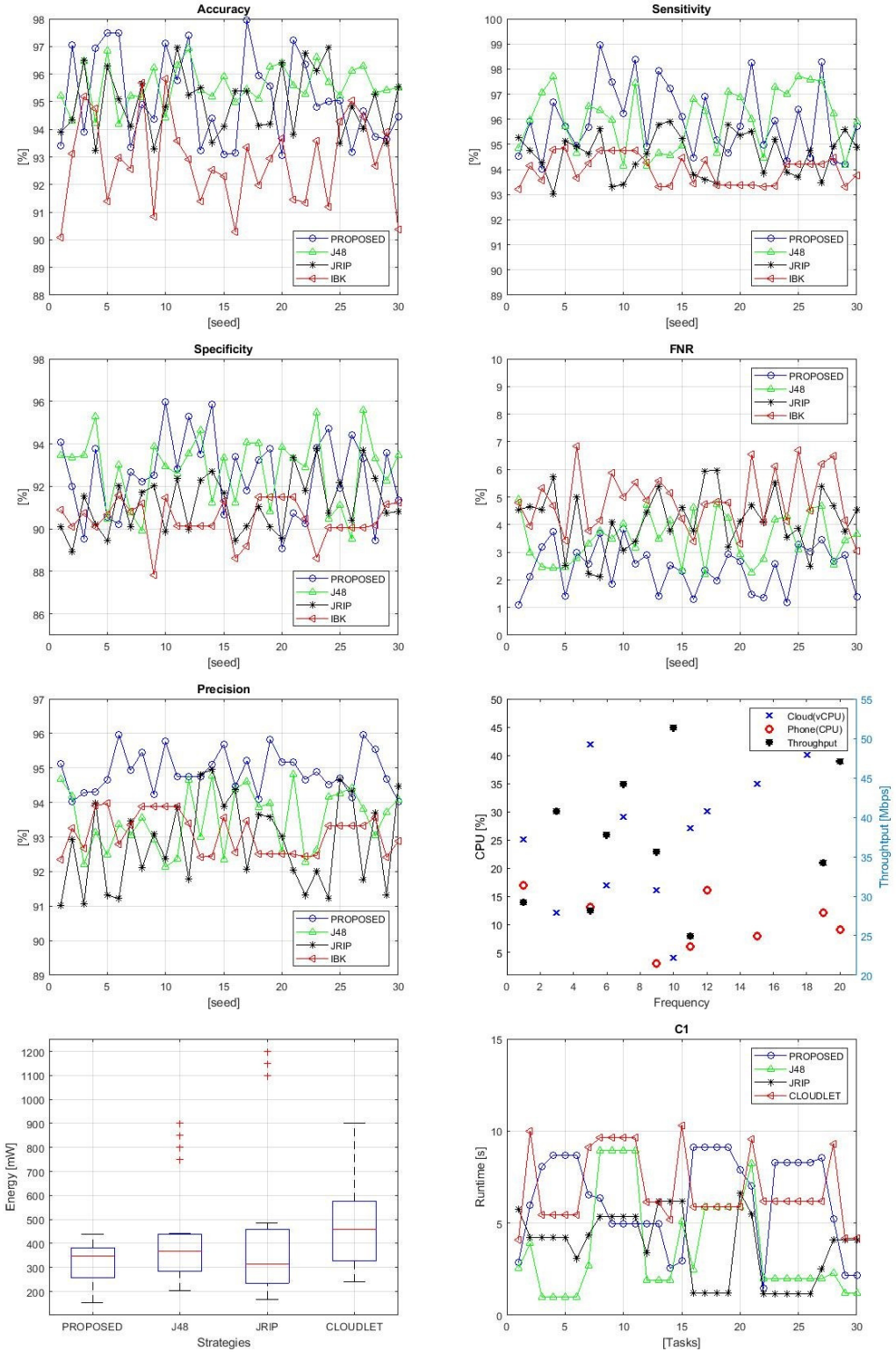


FIG. 5.1. *Comparison of the performances of the classification algorithms using different indicators – from left to right and from top to bottom: (a) Accuracy; (b) Sensitivity; (c) Specifity; (d) FNR; (e) Precision; (f) Throughput; (g) Energy consumption; (h) Execution time*

REFERENCES


[1] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, *MAUI: Making Smartphones Last Longer with Code Offload*, in ACM MobiSys 2010, Association for Computing Machinery, Inc., June 2010, https://www.microsoft.com/en-us/research/publication/maui-making-smartphones-last-longer-with-code-offload/.

[2] K. Akherfi, M. Gerndt, and H. Harroud, *Mobile cloud computing for computation offloading: Issues and challenges*, Applied Computing and Informatics, 14 (2018), pp. 1 – 16, https://doi.org/https://doi.org/10.1016/j.aci.2016.11.002, http://www.sciencedirect.com/science/article/pii/S2210832716300400.

[3] A. Al-Badarneh, M. Ali, and S. M. Ghaleb, *An Improved Classifier for Arabic Text*, Journal of Convergence Information Technology (JCIT), 11 (2016), pp. 69–84, http://www.globalcis.org/dl/citation.html?id=JCIT-4357&Search=&op=Title.

[4] A. Alelaiwi, *An efficient method of computation offloading in an edge cloud platform*, Journal of Parallel and Distributed Computing, 127 (2019), pp. 58 – 64, https://doi.org/https://doi.org/10.1016/j.jpdc.2019.01.003, http://www.sciencedirect.com/science/article/pii/S0743731519300140.

[5] X. Chen, *Decentralized Computation Offloading Game for Mobile Cloud Computing*, IEEE Transactions on Parallel and Distributed Systems, 26 (2015), pp. 974–983, https://doi.org/10.1109/TPDS.2014.2316834.

[6] X. Chen, S. Chen, X. Zeng, X. Zheng, Y. Zhang, and C. Rong, *Framework for Context-aware Computation Offloading in Mobile Cloud Computing*, J. Cloud Comput., 6 (2017), pp. 71:1–71:17, https://doi.org/10.1186/s13677-016-0071-y, https://doi.org/10.1186/s13677-016-0071-y.

[7] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, *CloneCloud: Elastic Execution Between Mobile Device and Cloud*, in Proceedings of the Sixth Conference on Computer Systems, EuroSys '11, New York, NY, USA, 2011, ACM, pp. 301–314, https://doi.org/10.1145/1966445.1966473, http://doi.acm.org/10.1145/1966445.1966473.

[8] Fei Gu and Jianwei Niu and Zhiping Qi and Mohammed Atiquzzaman, *Partitioning and offloading in smart mobile devices for mobile cloud computing: State of the art and future directions*, Journal of Network and Computer Applications, 119 (2018), pp. 83 – 96, https://doi.org/{https://doi.org/10.1016/j.jnca.2018.06.009}, {http://www.sciencedirect.com/science/article/pii/S1084804518302170}.

[9] D. Fesehaye, Y. Gao, K. Nahrstedt, and G. Wang, in 2012 IEEE 16th International Enterprise Distributed Object Computing Conference.

[10] S. Ghasemi-Falavarjani, M. Nematbakhsh, and B. S. Ghahfarokhi, *Context-aware multi-objective resource allocation in mobile cloud*, Computers and Electrical Engineering, 44 (2015), pp. 218 – 240, https://doi.org/https://doi.org/10.1016/j.compeleceng.2015.02.006, http://www.sciencedirect.com/science/article/pii/S0045790615000312.

[11] R. A. Hasan, M. A. Mohammed, Z. H. Salih, M. A. B. Ameedeen, N. ŢĂPUŞ, and M. N. Mohammed, *HSO: A Hybrid Swarm Optimization Algorithm for Reducing Energy Consumption in the Cloudlets*, TELKOMNIKA, 16 (2018), pp. 2144–2154, https://search.proquest.com/docview/2126486677?accountid=33993.

[12] R. A. Hasan, M. A. Mohammed, Z. H. Salih, M. A. B. Ameedeen, N. ŢĂPUŞ, and M. N. Mohammed, *HSO: A Hybrid Swarm Optimization Algorithm for Reducing Energy Consumption in the Cloudlets*, TELKOMNIKA, 16 (2018), pp. 2144–2154, https://search.proquest.com/docview/2126486677?accountid=33993.

[13] R. A. HASAN and M. N. MOHAMMED, *A Krill Herd Behaviour Inspired Load Balancing of Tasks in Cloud Computing*, Studies in Informatics and Control, 26 (2017), https://doi.org/10.24846/v26i4y201705, https://app.dimensions.ai/details/publication/pub.1099870816andhttps://sic.ici.ro/wp-content/uploads/2017/12/SIC_2017-4-Art.5.pdf.

[14] H. Jadad, A. Touzene, K. Day, and N. Alzeidir, *A cloud-side decision offloading scheme for mobile cloud computing*, International Journal of Machine Learning and Computing, 8 (2018), pp. 367–371.

[15] W. Junior, E. Oliveira, A. Santos, and K. Dias, *A context-sensitive offloading system using machine-learning classification algorithms for mobile cloud environment*, Future Generation Computer Systems, 90 (2019), pp. 503 – 520, https://doi.org/https://doi.org/10.1016/j.future.2018.08.026, http://www.sciencedirect.com/science/article/pii/S0167739X17326729.

[16] S. Kosta, A. Aucinas, Pan Hui, R. Mortier, and Xinwen Zhang, *ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading*, in 2012 Proceedings IEEE INFOCOM, March 2012, pp. 945–953, https://doi.org/10.1109/INFCOM.2012.6195845.

[17] M. D. Kristensen and N. O. Bouvin, *Scheduling and development support in the Scavenger cyber foraging system*, Pervasive and Mobile Computing, 6 (2010), pp. 677 – 692, https://doi.org/https://doi.org/10.1016/j.pmcj.2010.07.004, http://www.sciencedirect.com/science/article/pii/S1574119210000581. Special Issue PerCom 2010.

[18] Y. Kwon, H. Yi, D. Kwon, S. Yang, Y. Cho, and Y. Paek, *Precise execution offloading for applications with dynamic behavior in mobile cloud computing*, Pervasive and Mobile Computing, 27 (2016), pp. 58 – 74, https://doi.org/https://doi.org/10.1016/j.pmcj.2015.10.001, http://www.sciencedirect.com/science/article/pii/S1574119215001856.

[19] K. Lin, S. Pankaj, and D. Wang, *Task offloading and resource allocation for edge-of-things computing on smart healthcare systems*, Computers and Electrical Engineering, 72 (2018), pp. 348 – 360, https://doi.org/https://doi.org/10.1016/j.compeleceng.2018.10.003, http://www.sciencedirect.com/science/article/pii/S0045790617339174.

[20] C. M. S. Magurawalage, K. Yang, L. Hu, and J. Zhang, *Energy-efficient and network-aware offloading algorithm for mobile cloud computing*, Computer Networks, 74 (2014), pp. 22 – 33, https://doi.org/https://doi.org/10.1016/j.comnet.2014.06.020, http://www.sciencedirect.com/science/article/pii/S1389128614003193. Special Issue on Mobile Computing for Content/Service-Oriented Networking Architecture.

[21] T. Meng, K. Wolter, H. Wu, and Q. Wang, *A secure and cost-efficient offloading policy for Mobile Cloud Computing against timing attacks*, Pervasive and Mobile Computing, 45 (2018), pp. 4 – 18, https://doi.org/https://doi.org/10.

1016/j.pmcj.2018.01.007, http://www.sciencedirect.com/science/article/pii/S1574119216304308.

[22] M. A. Mohammed, R. A. Hasan, M. A. Ahmed, N. Tapus, M. A. Shanan, M. K. Khaleel, and A. H. Ali, *A Focal load balancer based algorithm for task assignment in cloud environment*, in 2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), June 2018, pp. 1–4, https://doi.org/10.1109/ECAI.2018.8679043.

[23] M. A. MOHAMMED and N. ŢĂPUŞ, *A Novel Approach of Reducing Energy Consumption by Utilizing Enthalpy in Mobile Cloud Computing*, Studies in Informatics and Control, 26 (2017), https://doi.org/10.24846/v26i4y201706, https://app.dimensions.ai/details/publication/pub.1099870815andhttps://sic.ici.ro/wp-content/uploads/2017/12/SIC_2017-4-Art.6.pdf.

[24] N. Q. Mohammed, M. S. Ahmed, M. A. Mohammed, O. A. Hammood, H. A. N. Alshara, and A. A. Kamil, *Comparative Analysis between Solar and Wind Turbine Energy Sources in IoT Based on Economical and Efficiency Considerations*, in 2019 22nd International Conference on Control Systems and Computer Science (CSCS), May 2019, pp. 448–452, https://doi.org/10.1109/CSCS.2019.00082.

[25] F. A. Nakahara and D. M. Beder, *A context-aware and self-adaptive offloading decision support model for mobile cloud computing system*, Journal of Ambient Intelligence and Humanized Computing, 9 (2018), p. 1561, https://doi.org/https://doi.org/10.1007/s12652-018-0790-7.

[26] C. Z. Radulescu and M. Radulescu, *Group decision support approach for cloud quality of service criteria weighting*, Studies in Informatics and Control, 27 (2018), pp. 275–284.

[27] Z. H. Salih, G. T. Hasan, M. A. Mohammed, M. A. S. Klib, A. H. Ali, and R. A. Ibrahim, *Study the Effect of Integrating the Solar Energy Source on Stability of Electrical Distribution System*, in 2019 22nd International Conference on Control Systems and Computer Science (CSCS), May 2019, pp. 443–447, https://doi.org/10.1109/CSCS.2019.00081.

[28] Shuang Chen, Yanzhi Wang, and M. Pedram, *A semi-Markovian decision process based control method for offloading tasks from mobile devices to the cloud*, in 2013 IEEE Global Communications Conference (GLOBECOM), Dec 2013, pp. 2885–2890, https://doi.org/10.1109/GLOCOM.2013.6831512.

[29] G. Skourletopoulos, C. X. Mavromoustakis, G. Mastorakis, J. M. Batalla, C. Dobre, S. Panagiotakis, and E. Pallis, *Towards Mobile Cloud Computing in 5G Mobile Networks: Applications, Big Data Services and Future Opportunities*, Springer International Publishing, Cham, pp. 43–62, https://doi.org/10.1007/978-3-319-45145-9_3, https://doi.org/10.1007/978-3-319-45145-9_3.

[30] H. Tout, C. Talhi, N. Kara, and A. Mourad, *Smart mobile computation offloading: Centralized selective and multi-objective approach*, Expert Systems with Applications, 80 (2017), pp. 1 – 13, https://doi.org/https://doi.org/10.1016/j.eswa.2017.03.011, http://www.sciencedirect.com/science/article/pii/S0957417417301586.

[31] F. Yu, H. Chen, and J. Xu, *DMPO: Dynamic mobility-aware partial offloading in mobile edge computing*, Future Generation Computer Systems, 89 (2018), pp. 722 – 735, https://doi.org/https://doi.org/10.1016/j.future.2018.07.032, http://www.sciencedirect.com/science/article/pii/S0167739X17329813.

[32] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, *A Context Sensitive Offloading Scheme for Mobile Cloud Computing Service*, in 2015 IEEE 8th International Conference on Cloud Computing, June 2015, pp. 869–876, https://doi.org/10.1109/CLOUD.2015.119.

[33] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, *mCloud: A Context-Aware Offloading Framework for Heterogeneous Mobile Cloud*, IEEE Transactions on Services Computing, 10 (2017), pp. 797–810, https://doi.org/10.1109/TSC.2015.2511002.

*Edited by:* Dana Petcu
*Received:* Jun 6, 2019
*Accepted:* Aug 9, 2019