Energy Efficient Ant Colony Cloud Offloading Algorithm (EACO)

Christina William Danial
Information Technology Department,
Faculty of Computers and Artificial Intelligence,
Cairo University, Egypt.
c.william@fci-cu.edu.eg

Iman Aly Saroit
Information Technology Department,
Faculty of Computers and Artificial Intelligence,
Cairo University, Egypt.
i.saroit@fci-cu.edu.eg

Shaimaa M. Mohamed
Information Technology Department,
Faculty of Computers and Artificial Intelligence,
Cairo University, Egypt.
s.mosaad@fci-cu.edu.eg

Abstract
Mobile cloud computing is a computing paradigm that helps to reduce the application energy consumption, so increases the battery life. Mobile application is divided to fine grained tasks with sequential and parallel topology. Offloading application tasks to cloud provides more energy but increases the completion time. The scheduling of tasks between executing in mobile device and cloud is more important to limit the increasing in the completion time. In this paper, energy efficient ant colony cloud offloading algorithm (EACO) is developed to reduce the energy consumption with the hard condition of completion time. EACO decreases the energy by an average 24%-59% with increasing in completion time by 3.6%- 28% compared with previous work. The algorithm is verified in different experiments with different tasks input data and computation workload.

CCS Concepts
•Networks→Network services→Cloud computing• Hardware→Power and energy• Networks→Network types→Mobile networks

Keywords
MCC; Ant colony algorithm; cloud computing

1. INTRODUCTION
Nowadays, mobiles are used extensively; numerous mobile applications are developed at each minute. These applications range from games and Internet applications to readers, images, and voices. In spite of all those incredible achievements, especially with the spread of the Internet technologies, mobile devices still have their limitations in terms of energy, storage and processing. Therefore, techniques are employed to save energy. One of these techniques is offloading to a cloud server.

Mobile cloud computing (MCC) is the integration between cloud computing and mobile environment to utilize resources in terms of on-demand resources [1]. MCC aimed at enhancing the mobile devices performance. So that, MCC extends powerful computation capability and the battery life, minimize the delay in executing mobile applications and so save energy for mobile devices. The main characteristic of MCC is the mobility where the application can continue regardless of user movement. MCC needs to offload computation intensive tasks from a mobile to the cloud via wireless access, So mobile devices don't need powerful hardware. One of challenges in MCC is the energy efficient task scheduling.

ICSIE 2020, November 11–13, 2020, Cairo, Egypt

ACM ISBN 978-1-4503-7721-8/20/04…$15.00
DOI: https://doi.org/10.1145/3436829.3436846

The energy efficient task scheduling depends on dividing the application into tasks. They are either processed on the mobile or offloaded to the cloud. The problem of determining which task to be offloaded is considered an NP-complete problem [2]. Typically, optimization techniques are used to get near-optimal solutions but they have some constraints. Recently, evolutionary optimization techniques, such as genetic algorithm, ant colony algorithms and others are used to avoid the constraints of the previous optimization techniques [3].

The prospective benefits of MCC can be achieved by considering the following questions: (1) How to achieve energy efficient scheduling for the parts of application? (2) Will the completion time be affected by reducing energy? (3) What is the optimization technique that will get near to optimal solution?
In this paper the previous questions are addressed. The proposed solution will be focused on achieving energy efficient ant colony cloud offloading algorithm (EACO) under hard constraint for application completion time. EACO decrease the energy by an average 24%-59% with increasing in completion time by 3.6%-28% compared to liu's work in [2].

The rest of the paper is organized as follows. Section two discusses the previous solutions for the offloading problem. Section three formulates the optimal task scheduling problem and explains the basics for the approach. The proposed solution EACO is discussed in section four. The results and analysis of EACO are in section five. Finally, the work is concluded in section six.

2. RELATED WORK
Efficient offloading techniques based on optimization protocols follow a number of steps: initially, they collect the experiment data for example, the input, output and computation workload values for the application. After that, they implement the selected approach and finally evaluate the approach by calculating some performance metrics such as task completion time, energy. One of optimization techniques is classical approaches which are defining

an objective function subject to some constraints in order to reach the near-optimal solution. Nevertheless it has limited scope in practical applications as some of them involve continuous or differentiate objective function.

Authors in [4] employed the classical approach to save energy; they prepared the parameters by implementing feature-based prediction and evaluated the best prediction by implementing real experiments using the N-queen problem. The paper decreased the execution time by up to 31.7% compared to previous methods and saved the energy of smartphone by up to 57.2%.

The classical approach is the basis for developing most of the numerical techniques, it leads to a set of nonlinear simultaneous equations that may be difficult to solve, and thus the numerical techniques are the most used ones. Numerical techniques such as linear, integer, quadratic, nonlinear, stochastic, dynamic, combinatorial, infinite-dimensional and constraints satisfaction are applied in the offloading problem to find a near-optimal solution [5,6,7].

The graph based approach has been used extensively for solving the optimization problem. The graph-based approach was used in [8], where the authors used a minimum cut algorithm and a preflow push algorithm to calculate the residual network. They reported the effects of workload amount, network type, computation cost, security operations, signal strength and call graph structure on the optimized overall energy consumption [8]. Authors in [9] implemented the graph approach where they collected data using a historical-based approach. The authors employed minimum cut algorithm and made some updates for the graph at runtime. Additionally, in [2], the application was divided into tasks and an acyclic directed graph was drawn. The author applied the Lagrangian multiplier to the graph.

These above approaches can save the energy at the cost of performance, but there is still a need for faster approach with less problem knowledge and adaptable with changes [3,10,11]. Evolutionary approaches (EA) are developed for large scale optimization problems such as complex problems with discontinuous and possibly noisy target functions [11]. EA is straighting forward to apply, it provides a number of potential solutions and used for real applications. Examples of EAs are genetic algorithms, particle swarm optimization and ant colony algorithm. Genetic algorithm is used in [12] with a cooperation with machine learning to offload the application parts. Ant colony algorithm (ACO) is the most successful and widely recognized algorithmic based on ant behavior [13] and is used in the network field. Ant system is the original of ACO and the two other variants: Max-Min Ant system and Ant colony system. ACO can be run continuously and can adapt to changes in real time compared by the others techniques.

Authors in [14] used ACO to offload tasks on cloudlets. Cloudlets send the tasks to mobile devices that are willing to share their resources. The objective function of paper [14] was maximizing the profit of the execution without calculated the energy consumption. Cloudlets contain few servers with less powerful than the datacenter cloud [14], so datacenter cloud is preferred in the loaded and critical applications. In [15] the tasks offloaded using ACO, the used communication model depend on Gilbert-Elliot model that considers the channel switches between bad state and good state by a given probability. In [16] ACO is used for queue scheduling in cloudlets which minimized the completion time. In [17] ACO is used to schedule the services of clients to be processed in multiple cloud servers. Authors' goal was minimizing the service time; they did not consider the energy consumption nor the channel model. The proposed technique focused on services running on the mobile and did not consider partitioning the applications. To the best of our knowledge, this is the first time to use ant colony algorithm for offloading application to the cloud datacenter. EACO offloads the application in cloud datacenter not cloudlet, which can be used in heavy applications because cloudlet has small power. EACO obtains efficient transmission rate using IEEE 802.11 standards in communication model instead of Gilbert-Elliot model, that only considers the channel switches between bad state and good state by a given probability. EACO goal is to minimize the energy consumption with considering the hard constraint of completion time of the application.

3. PROBLEM FORMULATION

EACO consists of three models; First, task model which defines the sequence of tasks as parallel or series tasks. The second model is the communication one which determines the data rate at which tasks are sent to the cloud. The Third model draws the graph and determines the costs of the edges that connect the nodes.

3.1 Task Model

Mobile device applications are divided into fine-grained tasks. The simplest sequence of tasks is a series in which the output of the task is the input for the next one. However, parallel tasks are more real where tasks processed at the same time. The inputs for parallel tasks are the same but each task have different output, and thus, the outputs should be combined in order to be an input for the next task. EACO algorithm follows the task model in [2] where series and parallel tasks are combined.

3.2 Communication Model

This model determines the data rate of the communication channel. The proposed algorithm follows the same model in [2] that gives expectation value for the data rate (Rt). This data rate affects the transmission time and the energy consumption of the mobile device. The model depends on the characteristics of distributed coordination function (DCF) mechanisms in IEEE 802.11 standards. The parameters needed in the model are mentioned in Table 1. Equations and details of the model can be found in [2].

3.3 Execution Model

This model is used to calculate the energy consumption and completion time for each task. They depend on idle power (Pi), reception power (Pr), transmitting power (Pt)p_t and execution power (Pm). Eight parameters are calculated in the

mobile device using the four power parameters; energy of mobile(Em), time for processing in mobile (Tm), cloud energy(Ec), time for processing in cloud (Tc), energy (Erm) and time (Trm) for transfer data from cloud to mobile and energy (Eoc) and time (Toc) for transfer data from mobile to cloud. Consequently, to determine the calculated parameters for each task, the processing of the preceding task should be determined. The parameters are calculated using the model in [2]. The energy parameters are added to each other in order to determine the first cost of the edge of the graph. The time parameters are added to be the second cost of the edge. The rules of making the addition explained in graph model.

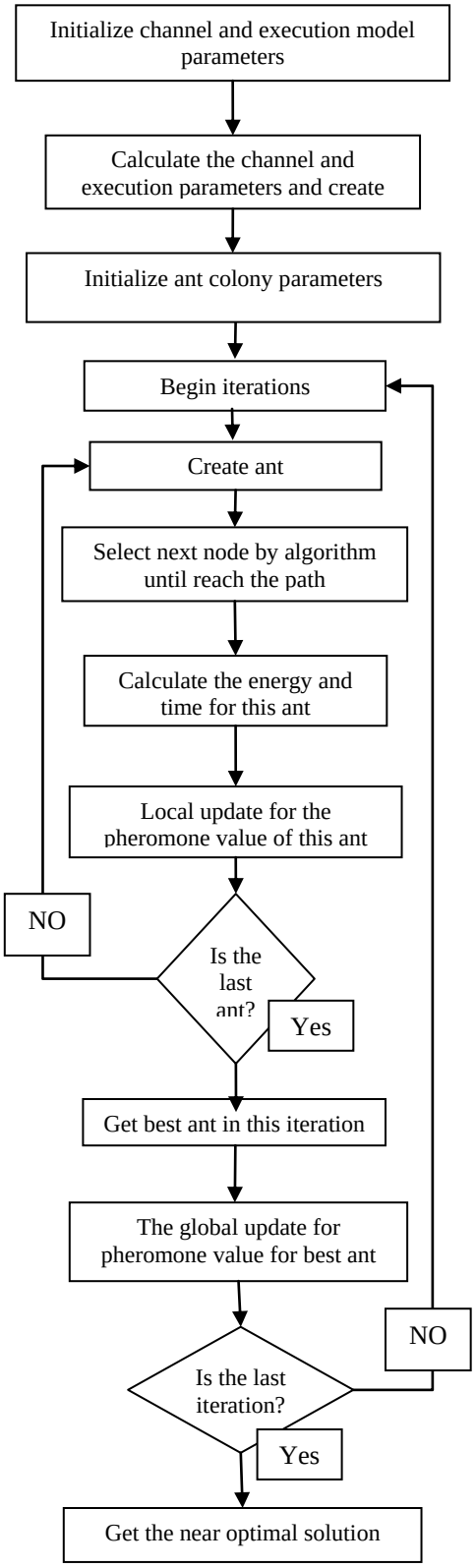

Figure 1. Flow chart of EACO

3.4 Graph Model

Acyclic directed graph model is used to represent the energy efficient problem. Tasks are represented as nodes while the energy and completion time as cost for each edge in the graph. Therefore, every step u, has some tasks with two options processed in mobile or in a cloud. Parameter C is put to indicate the decision of offloading when offloading to the cloud $C=1$ otherwise $C=0$. The costs for each edge are energy consumption and completion time. The equations of energy and completion time for the edges are shown in [2].

3.5 Proposed Algorithm

In this section, the objective function is defined and the proposed algorithm is discussed. Since, there are a lot of paths to follow in the graph, finding the optimal path is considered an NP-complete problem [2]. The optimal path is found by choosing the minimum energy consumption path with reducing the increasing in completion time. The objective function (Eq. 1) is formulated such that the minimum energy can be achieved given that the completion time does not exceed the threshold time.

$$\min \sum E_{u-1,u}$$

s.t.

$$\sum T_{u-1,u} \leq T_d \quad (1)$$

Where $E_{u-1,u}$ is the energy of the edge that consumes when move from step u-1 to step u and $T_{u-1,u}$ is the completion time of step u depends on step u-1.

Ant Colony system is evolutionary approach [13], [18] which is used for solving NP-complete problems. EACO aims to get the near-optimal path that minimizes the energy without affecting the completion time. A flowchart for the proposed algorithm is shown in Fig. 1.

The Ant Colony System is used to search for an optimal path in the graph. Each ant moves along a path where it leaves a pheromone. The ant represents a thread from the application, where each node is a task where EACO choose the thread or path with the minimum energy.The pheromone value is represented by V. The ant moves from one node to another by applying the pseudo-random proportional rule. The next node j is chosen by Eq. 2:

$$j = \begin{cases} \max_{u \in Kj} V_u^{\alpha} \frac{1}{E_U}^{\beta} \frac{1}{T_u}^{\gamma} & if \quad m \leq m_o \\ J & otherwise \end{cases} \quad (2)$$

Where thepath maximizes the pheromone value V and minimizes the energy E and completion time T is chosen. The parameter α is the pheromone importance, β is the energy importance and γ is the completion time importance. Variable m is a random number uniformly distributed in the range [0, 1]. Parameter m_o is a specified number in the range [0, 1] and the Kj is the possible paths. If $m \prec m_o$ the next node will be determined by Eq. 3:

$$P_u = \begin{cases} \frac{V_u^{\alpha} \frac{1}{E_u}^{\beta} \frac{1}{T_u}^{\gamma}}{\sum V_u^{\alpha} \frac{1}{E_u}^{\beta} \frac{1}{T_u}^{\gamma}} & if\ u \in Kj \\ 0 & otherwise \end{cases} \quad (3)$$

The probability for all possible next nodes is calculated by this equation, when the node with minimum probability is chosen. After the ant creates the path, the proposed algorithm calculates the energy and completion time for the path and updates the pheromone for this path by local update in Eq. 4:

$$V_u = (1-\rho)V_u + \rho V_o \quad (4)$$

where evaporation ratio ρ specifies a number from range [0-1] and Initial pheromone V_o put for all paths before beginning the iterations. V_o is calculated by the following Eq. 5:

$$V_o = (nE_t) - 1 \quad (5)$$

where n is the number of steps in this application and E_t is the total energy for execution in a mobile device. Then the algorithm performs these steps for a number of ants then chooses the best ant that achieves the objective function, so update the pheromone for this path. This update is called global update , it is applied by the given Eq. 6:

$$V_u = (1-\rho)V_u + \rho\, {}^{1}/_{n} \quad (6)$$

After repeating the steps for some iteration, the ant with the maximum pheromone value, i.e. which achieves the objective function is chosen.

4. SIMULATION AND ANALYSIS

This section presents the simulation results of EACO. In order to evaluate the performance of the proposed scheduling mechanism, EACO are compared with liu's model in [2] where the experiments of liu have been tried with EACO. EACO's experiments are run on a laptop with 2.5 GHz AMD CPU, 4GB of RAM, Microsoft Win8 operating System. The algorithm is implemented in Java. The algorithm is tested in 5 applications from liu's work with different computation workload and different input data in order to determine for each task the decision of offloading then calculate the energy consumption, the completion time and the percentage of reduction. The computation workload is the computed amount of work in cycles. The input data is the data in bits for each task that will be offloaded in case of execute in cloud.

Ant Colony System has more parameters that vary from a problem to another one, so EACO makes exhaustive search to get the optimal values that gives the best decision of offloading mentioned in Table 1. The application is divided into 10 steps with 13 tasks as shown in Fig.2 which is the same for liu [2].

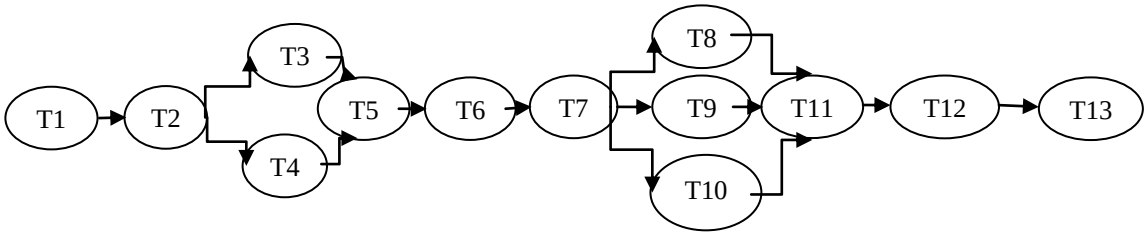

Figure 2. The sequence of tasks

Table 1.ant colony parameters

Parameter	Value
A	Random number [3-150]
B	Random number [3-50]
γ	Random number[1-200]
m_o	0.9
P	0.1
Number of ants	34
Number of iterations	1-400

The following charts show the experiments results. For each one inputs are represented with dashed bar,while computation workloads are represented with solid bar. The bars with three parts mean that this step has three tasks with three inputs and three workload such as step 7 in Fig.3. The decision of offloading is represented by line on the top of the graph where each task executed in the mobile be at value 0 and the one executed in cloud be at value 1.

As shown in Fig.3 and Table 2 of experiment1, when the used threshold of completion time is 1.5 second, the energy is reduced by 26% but the completion time is increased by 23% when that of Liu's algorithm. The negative sign in time reduction is put to indicate that this value increased. Consequently, the need for reducing the value of completion time is important, sois reduced to 0.7, this decreases the completion time with a little increase in the energy consumption. The result is shown in Table 3 and Fig.4.

Table 2. Experiment1 values Td=1.5

Metric	Liu's work	Ant colony
Energy	0.05 J	0.037 J
Completion time	0.608 s	0.79 s
Energy reduction		26%
Time reduction		-23%

Table 3. Experiment1 values Td=0.7

Metric	Liu's work	Ant colony
Energy	0.05 J	0.038 J
Completion time	0.608 s	0.435 s
Energy reduction		24%
Time reduction		28%

Table 4. Experiment2 values

Metric	Liu's work	Ant colony
Energy	0.0598 J	0.0598 J
Completion time	0.404 s	0.404 s
Energy reduction		0%
Time reduction		0%

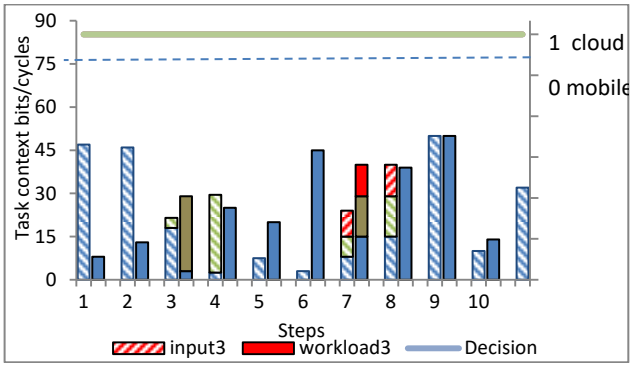

Figure 3.The decision of experiment1 with Td=1.5

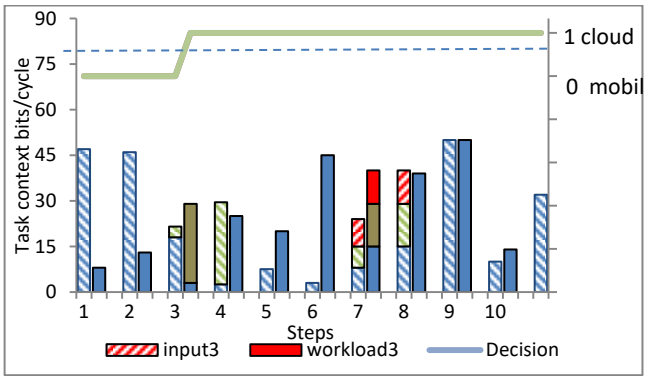

Figure 4.The decision of experiment1 with Td=0.7

In experiment2, no change occurs after applying the Ant Colony. Since, the standard deviation of computation workload is 18.925 which is a big value,the completion time of offloading to cloud is big so that no change occur. The graph and results of experiment 2 are shown i

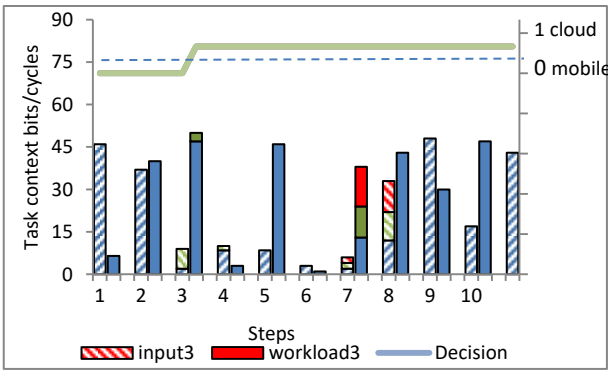

Figure 5. The decision of experiment2

In experiment 3, Liu tries two values for Td=1.5 and 3, the two have the same end result. When Td=1.5 reduction occurs but when Td=3 nochange occurs. The results of the two cases are shown in Fig 6, Table 5 and Fig 7, Table 6 respectively.

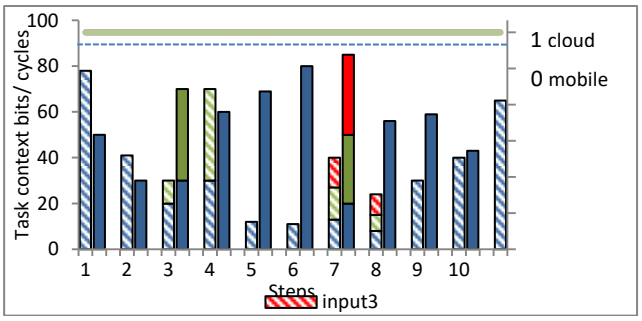

Figure 6.The decision of experiment3 Td=1.5

Table 5. Experiment 3 values Td=1.5

Metric	Liu's work	Ant colony
Energy	0.0799 J	0.057 J
Completion time	0.79 s	1.255 s
Energy reduction		28%
Time reduction		-37%

Table 6. Experiment 3 values Td=3

Metric	Liu's work	Ant colony
Energy	0.0498 J	0.0498 J
Completion time	1.098 s	1.098 s
Energy reduction		0%
Time reduction		0%

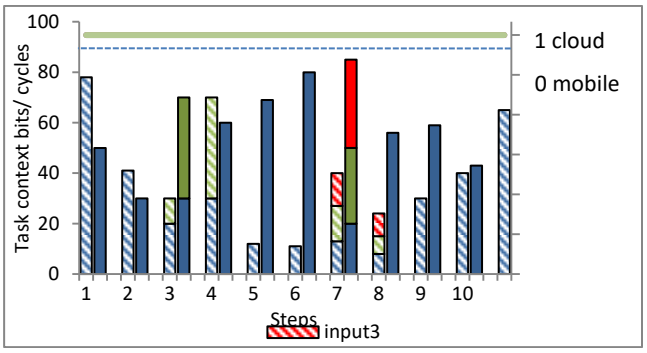

Figure 7.The decision of experiment3 Td=3

In experiment 4, Liu tries three values for Td 0.5,1.5, 3. If Td=0.5 no change occurs otherwise reduction occurs as shown in Fig 8, Table 7, Fig 9, Table 8 , Fig 10 and Table 9.

Table 7. Experiment 4 values Td=0.5

Metric	Liu's work	Ant colony
Energy	0.176 J	0.176 J
Completion time	0.352 s	0.352 s
Energy reduction		0%
Time reduction		0%

Table 8. Experiment 4 values Td=1.5

Metric	Liu's work	Ant colony
Energy	0.0707 J	0.0288 J
Completion time	0.582 s	0.604 s
Energy reduction		59%
Time reduction		-3.6%

Table 9. Experiment 4 values Td=3

Metric	Liu's work	Ant colony
Energy	0.0707 J	0.0288 J
Completion time	0.582 s	0.604 s
Energy reduction		59%
Time reduction		-3.6%

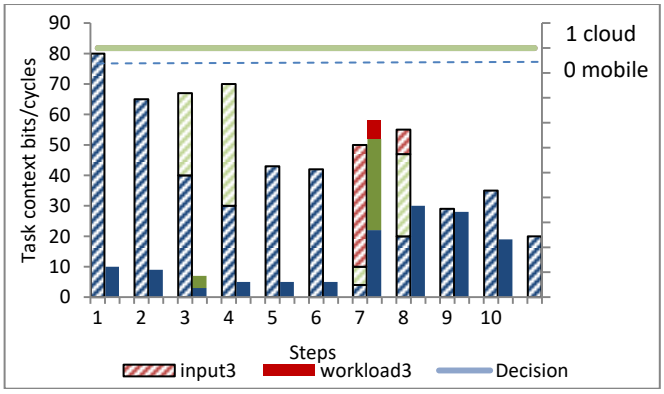

Figure 8.The decision of experiment 4 Td=0.5

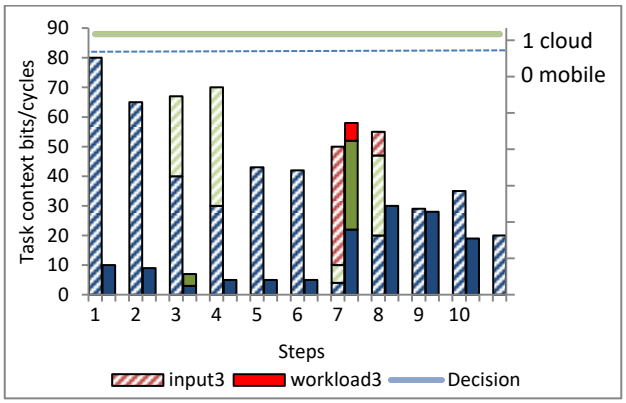

Figure 9. The decision of experiment4 Td=1.5

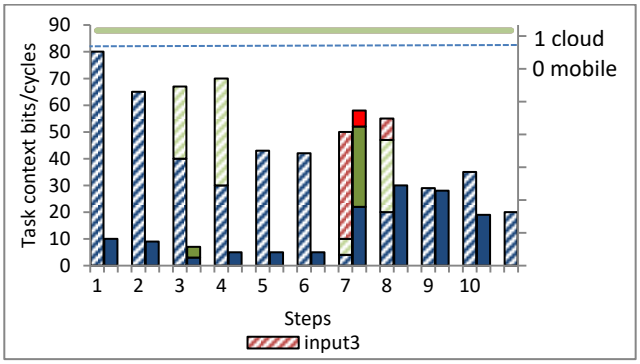

Figure 10. The decision of experiment 4 Td=3

In experiment 5, three values are tried Td=0.5, 1.5, 3. If T_d=0.5 no change happens,but in the other two cases the energy is decreased by 52 % and the completion time is increased by small value 5.6 seconds. This is made clear in the Figures (11, 12, 13)and tables (10, 11, 12).

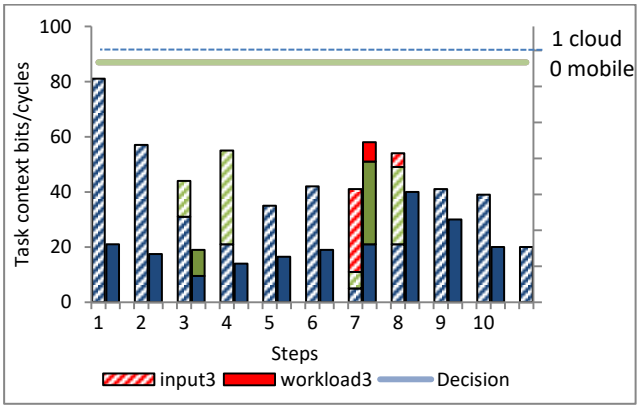

Figure 11. The decision of experiment5 Td=0.5

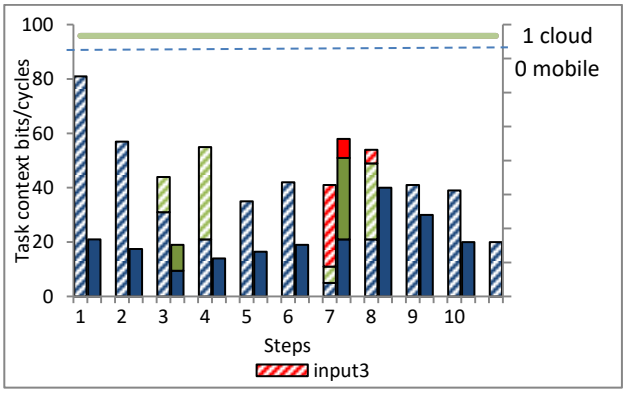

Figure 12. The decision of experiment5 Td=1.5

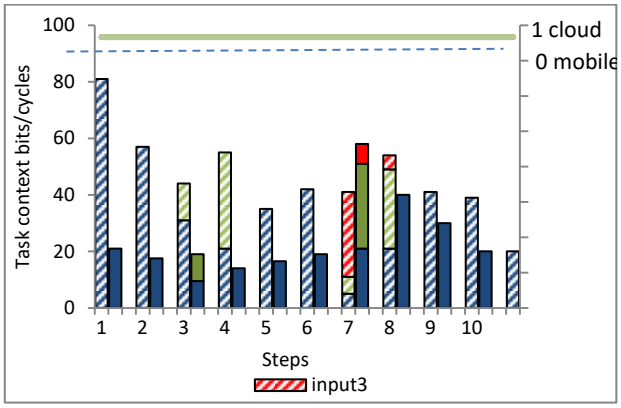

Figure 13. The decision of experiment5 Td=3

Table 10. Experiment 5 values Td=0.5

Metric	Liu's work	Ant colony
Energy	0.255 J	0.255 J
Completion time	0.51 s	0.51 s
Energy reduction		0%
Time reduction		0%

Table 11. Experiment 5 values Td=1.5

Metric	Liu's work	Ant colony
Energy	0.0805 J	0.038 J
Completion time	0.77 s	0.816 s
Energy reduction		52%
Time reduction		-5.6%

Table 12. Experiment 5 values Td=3

Metric	Liu's work	Ant colony
Energy	0.0805 J	0.038 J
Completion time	0.77 s	0.816 s
Energy reduction		52%
Time reduction		-5.6%

Finally, we can say that the Ant Colony System gets a better solution than Liu's work in terms of energy but at the expense of increasing the completion time. The data is analyzed to see what increases the completion time. The computation workload is the main parameter effect on the decision, and so the standard deviation for experiments is calculated. In experiment 2, there is no change because the standard deviation of computation workload 18.925 which is a big value. The standard deviations of experiments 3 and 1 are 17.735,14.6 respectively because of that the completion time increases by moderated value. Consequently, experiments 4 and 5 increase the completion time by small value as the standard deviations are 10.65 and 9.33, respectively. EACO decreases the energy by an average 24%-59% with increasing in completion time by 3.6%- 37% compared to liu's work. Table 13 shows the comparison between EACO and liu based on the average energy consumption and the average completion time for each threshold. In our opinion EACO get better solution because the ant colony algorithm gets near optimal solution better than the shortest path algorithm that used in liu's work.

EACO gives a better solution especially if the computation workload of the experiment is small. The correct decision has been taken to reduce the energy despite the increases in completion time.

Table 13. Comparison between Liu and EACO results

	Average Energy consumption (J)		Average completion time (s)	
	Liu	EACO	Liu	EACO
Td=0.5	0.135	0.132	0.468	0.425
Td=1.5	0.068	0.0441	0.63	0.773
Td=3	0.067	0.038	0.816	0.839

5. CONCLUSION AND FUTURE WORK

In this paper, EACO tried to find the near optimal decision for the execution of application tasks between the mobile device and the cloud server. The main objective function is reducing the energy consumed on the mobile devices which increasing the battery life time. The completion time consider is the main constraint where we needs to reduce the energy consumption without increasing in completion time. EACO uses the ant colony algorithm to solve this problem and find the near to optimal decision.

EACO uses the same experiments and the same topology of tasks of liu's work. EACO decreases the energy by an average 24%-59% with increasing in completion time by 3.6%- 37% compared to liu's work.

In future work other experiments with different inputs and workload will be used. Ant colony system is used in this paper, in future other Evolutionary optimization approaches will be used.

6. REFERENCES

[1] Rahimi, M. Reza, Jian Ren, Chi Harold Liu, Athanasios V. Vasilakos, and NaliniVenkatasubramanian. "Mobile cloud computing: A survey, state of art and future directions." *Mobile Networks and Applications* 19, no. 2 (2014): 133-143.

[2] Liu, Tundong, Fufeng Chen, Yingran Ma, and Yi Xie. "An energy-efficient task scheduling for mobile devices based on cloud assistant." *Future Generation Computer Systems* 61 (2016): 1-12.

[3] Mishra, Surendra. "Optimization studies for physics problems in indian PHWRs." (2012).

[4] Kwon, Yongin, Hayoon Yi, Donghyun Kwon, Seungjun Yang, Yeongpil Cho, and YunheungPaek. "Precise execution offloading for applications with dynamic behavior in mobile cloud computing." *Pervasive and Mobile Computing* 27 (2016): 58-74.

[5] Guo, Songtao, Bin Xiao, Yuanyuan Yang, and Yang Yang. "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing." In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pp. 1-9. IEEE, 2016.

[6] Liu, Kaiyang, Jun Peng, Heng Li, Xiaoyong Zhang, and Weirong Liu. "Multi-device task offloading with time-constraints for energy efficiency in mobile cloud

computing." *Future Generation Computer Systems* 64 (2016): 1-14.

[7] Terefe, Mati B., Heezin Lee, NojungHeo, Geoffrey C. Fox, and Sangyoon Oh. "Energy-efficient multisite offloading policy using Markov decision process for mobile cloud computing." *Pervasive and Mobile Computing* 27 (2016): 75-89.

[8] Saab, Salwa Adriana, Farah Saab, Ayman Kayssi, Ali Chehab, and Imad H. Elhajj. "Partial mobile application offloading to the cloud for energy-efficiency with security measures." *Sustainable Computing: Informatics and Systems* 8 (2015): 38-46.

[9] Kaya, Mahir, AltanKoçyiğit, and P. ErhanEren. "An adaptive mobile cloud computing framework using a call graph based model." *Journal of Network and Computer Applications* 65 (2016): 12-35.

[10] Streichert, Felix. "Introduction to evolutionary algorithms." *paper to be presented Apr* 4 (2002).

[11] Pohlheim, Hartmut. "Evolutionary algorithms." GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with MATLAB Documentation. GEATbx (2003).

[12] Jin, Xiaomin, Zhongmin Wang, and Wenqiang Hua. "Cooperative Runtime Offloading Decision Algorithm for Mobile Cloud Computing." *Mobile Information Systems* (2019).

[13] Katiyar, Sapna, N. Ibraheem, and Abdul Quaiyum Ansari. "Ant colony optimization: a tutorial review." In Proceedings of the National Conference on Advances in Power and Control, ManavRachna International University, Faridabad, Haryana, India, vol. 574. 2015.

[14] Wei, Xianglin, Jianhua Fan, Ziyi Lu, and Ke Ding. "Application scheduling in mobile cloud computing with load balancing." *Journal of Applied Mathematics* (2013).

[15] Wang, Tongxiang, Xianglin Wei, Chaogang Tang, and Jianhua Fan. "Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints." *Peer-to-Peer Networking and Applications* 11, no. 4 (2018): 793-807.

[16] Sundararaj, Vinu. "Optimal task assignment in mobile cloud computing by queue based ant-bee algorithm." *Wireless Personal Communications* 104, no. 1 (2019): 173-197.

[17] Bao, Weidong, Haoran Ji, Xiaomin Zhu, Ji Wang, Wenhua Xiao, and Jianhong Wu. "ACO-based solution for computation offloading in mobile cloud computing." *Big Data & Information Analytics* 1, no. 1 (2016): 1.

[18] Nanda, Bijaya Kumar, and Gyanesh Das. "Ant colony optimization: a computational intelligence technique." *Int. J. Comput. Commmun. Technol* 2, no. 6 (2011): 105-110.