Original Article

Resource Sharing Model for Mobile Cloud Computing Network Users Based on Artificial-Bee-Colony Algorithm

Naimeh Sharifi

Department of computer and Information Technology, Faculty of Computer and Information Technology, University of Hadaf Institute of Higher Education, Sari, Iran.

Abstract

Mobile cloud services are distributed infrastructures creating communication and service. There are two levels of scheduling in cloud computing. The first level of scheduling tasks was on virtual machines (VMs) and the second level of scheduling VMs on servers. The focus of the proposed approach was on the second level that is the scheduling of VMs on physical machines. The most significant parameters in the proposed method were the memory and CPU demand of the VMs from the servers. Regarding this, the artificial bee colony (ABC) algorithm was used to allocate resources to requests. ABC algorithm is an optimization method induced by the behavior of bee processes. Finally, the purpose of the study was to reach the best load balancing in the cloud environment using optimization of CPU and memory consumption. Three methods - genetic algorithm (GA), firefly algorithm (FFA), and Bird Swarm Algorithm (BSA) - were used to compare the proposed method. The results showed that the proposed method has a better load balancing compared to the individual methods, and the total running time is less than the other methods in the proposed method.

Keywords: Resource sharing, cloud computing, ABC algorithm, FFA

Introduction

The increasing popularity and of the Internet and familiarity of the public with it throughout the world have led to the ever-increasing of the Internet services presentation. Moreover, it had made a large volume of marketing and e-commerce dependent on this infrastructure. As a platform, cloud presents different services to users regardless of location and hardware in return for a fee. Cloud computing has received great attention given the easier and faster development of web-based services at the universities and research centers [1].

Cloud includes infrastructure, software, platform, and other computing resources using a network. Cloud computing is a combination of two words: computing and cloud. Cloud refers to a vast network like the Internet where the lay user is unaware of the backstage and what goes on next. The graphs of computer networks use the cloud to show the Internet network. The reason for using the Internet infrastructure in the cloud is that it hides the technical details of the Internet from the users and creates a layer of abstraction between these technical details and the users [2].

Some studies have been conducted in this regard. Banerji et al. [2] have presented a dynamic energy-aware cloudlet-based mobile cloud computing model (DECM) that uses cloudlet techniques to allocate, manage, and optimize cloud-based productivity infrastructure and green computing access services. The model uses dynamic programming to collaborate with cloudlet resources by changing the operating environment. The purpose of DEC is adapting the practical needs of the mobile industry according to several factors affecting the quality of cloud service. Dasguptaa et al. [3] proposed a GA-based approach for load balancing in cloud computing. This algorithm performs the load on the cloud infrastructure by adjusting the size of the work set. Then it compares the proposed algorithm with First in first out (FIFO) and hill climbing algorithms. The results show that the improvement of the genetic method compared to the two methods mentioned. Khanchi et al. [4] proposed an algorithm by combining Active VM. Load Balancing (AVLB) and throttled load balancer (TVLB) algorithms to provide an algorithm for uniform distribution of workload among VMs.

Address for correspondence: Naimeh Sharifi, Department of computer and Information Technology, Faculty of Computer and Information Technology, University of Hadaf Institute of Higher Education, Sari, Iran.
Email: Info@hadaf.ac.ir

How to cite this article: Sharifi, N. Proposing a Resource Sharing Model for Users to Use on a Mobile Cloud Computing Network Based on the Bee Algorithm. Arch Pharma Pract 2020;11(S4):154-9.

The purpose of mobile cloud computing is to present a fast and secure approach for users to access information in the cloud via mobile devices. The main challenge of the system comes from the characteristics of mobile devices and wireless networks and their inherent faults. The limitations of mobile devices, the quality of wireless communications, the variety of applications and cloud computing support are all significant elements complicating designing, programming, and deployment of applications on mobile and distributed devices over fixed cloud devices that make it more efficient and affect using mobile cloud computing [5]. Dhinesh Babu and Venkata Krishna [6] have proposed an approach based on three architectures - single-user, multi-user - single virtual-machine and multi-user line to optimize the resources. Two problems have been used to test architecture. The first architecture calculates 10,000,000 random numbers and returns the final number.

Patel et al. [7] concluded that the cloud environment is a common pool of computing resources distributed across multiple computers depending on the type and amount of work. The proposed approach is an algorithm that works dynamically and examines the parameters performance increase, decrease communication costs, and optimization of resources. Kao et al. [8] have proposed an approach to optimize the minimum latency - Hermes algorithm. Kovachev et al. [9] have presented a method for ease of program development. Kokiko framework uses Android's inter-process communication (IPC) mechanisms for code dumping. Rehman Khan et al. [10] have introduced a system to reduce energy consumption by discharging fine-grained codes. In this system, programing methods can be removed using annotation. Xu and Fortes [11] have used a genetics-based algorithm where VMs are reconfigured in data centers with heterogeneous nodes, finding the optimal location for VMs according to environmental requirements.

Arian et al. [12] presented a new approach to multi-criteria cloud resource management and VM migration to decide on the load available. Studies have shown that one of the biggest problems with mobile cloud networks is the increase in service quality. The quality of service can be improved by optimizing the tasks available on the network. Selecting a proper optimization algorithm can have an effective role in enhancing service quality. This study has three general stages. The first stage is the necessary research in the field, the second is the implementation and coding of the research and the third is the recording of the test results. In this study, we try to present an optimal way to allocate and share resources in mobile cloud networks using the ABC optimization algorithm to increase service quality.

Methodology

The purpose of the study was to reach better resource sharing using load balancing in the mobile cloud. Thus, it was tried to enhance load balancing using the proposed algorithms to increase the efficiency of the cloud network. Figure 1 shows this model.

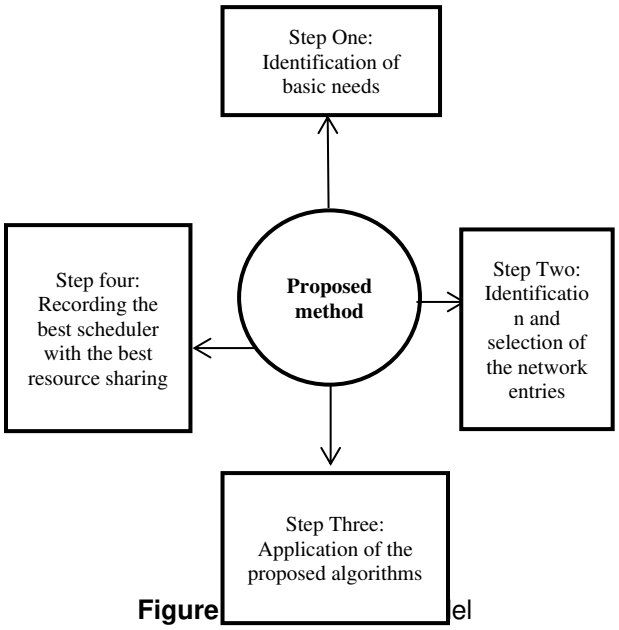

Figure el

ABC algorithm

The basic steps of the algorithm are:

- Initialization
- Iteration

A) The location of working bees in food resources in memory
B) The location of the bees in search of food resources in memory
C) Sending leading bees to search for new food sources

- Until obtaining the desired status

At the first step, ABC randomly distributes the initial population P (G = 0) SN solutions (food source positions), where SN shows the population size.

Each solution (food resource) xi (i = 1, 2,..., SN) is D-dimensional vector. Here, D is the number of optimization parameters. After initialization, the population of situations (solutions) is subjected to a cycle iteration, C = 1, 2,. . . , Cmax؛ where C is the search process for worker bees, scout and leaders.

ABC algorithm uses four various selection processes:

1. The process of global selection by artificial viewing bees to explore promising areas
2. a local selection process in the area is done by artificial worker bees and onlookers according to local information (about the real bees: this information includes color, shape, and aroma of flowers) (bees cannot identify the source of nectar until they reach the right place and discriminate between growing sources based on their aroma) to determine a neighbor's food resource around the memory source.
3. The local selection process, the greedy selection process, is done by all bees, where if the nectar value of the candidate resource is better than the current one, it

forgets the actual bee and retains the candidate resource. Otherwise, it keeps the current bee in memory.

4. A random selection process is done by the scout bee.

The above explanation shows three control parameters used in the main ABC:

- The number of food resources equal to the number of worker bees or spectator bees (SN)
- The value of the limit
- Maximum Number of Cycle (MCN)

Research formulation

In cloud computation-based data centers, there are sets of physical machines as clusters that have become fully virtualized. VM requests are entered in the amount of CPU and memory needed, and if accepted, they can be accommodated and deployed on these physical servers and otherwise rejected. Thus, the problem can be described as follows:

"Set I is composed of m VM $i \in I$and set J of n physical machines $j \in J$. Our goal is to deploy these VMs on the physical machines in the data center so that the total power consumption and thus the number of active physical machines and the server load balancing are minimized." The assumptions of the problem are as follows: first, all physical machines and VMs are shown by two processor and memory properties. The second assumption, we ignore the size of the disk assuming network-attached storage (NAS). The third assumption, no VM needs more resources than those provided on a physical server. The fourth assumption is that VMs are created according to the customer's request and according to its order. The fifth assumption is that the customers cannot order a request in any proportion of resources. The parameters of the problem are as described in Table 1.

Table 1: Problem parameters

Parameter	Description	Parameter	Description
I	VMs set	J	Physical machines set
m	Number of VMs	n	Number of physical machines
i	VM index	j	Physical machine index
T_j^{CPU}	Total CPU resource in j-th VM	A_j^{CPU}	Total CPU resource in j-th physical machine
T_j^{RAM}	Total memory resource in the j-th physical machine	A_j^{RAM}	Memory resource in the j-th physical machine
R_i^{CPU}	i-th VM processor request	R_i^{RAM}	i-th VM processor request
LB_j	J-th physical machine working load	LB_{DC}	Total load consumed in the data center
LB_{Avg}	Average data center consumed load		
U_j^{CPU}	The ratio of the CPU used to the total CPU available in the j-th physical machine		
X_{ij}	Binary variable, i-th VM deployment / non-deployment on j-th physical machine		
Y_j	Binary variable, use/non-use of the j-th physical machine		

$$LB_{DC} = \sum_{j=1}^{m} LBj \quad (1)$$

$$LB_{Avg} = \frac{LB_{DC}}{j} \quad (2)$$

LB_{Avg} Minimize (3)

$$\sum_{j=1}^{m} X_{ij} = 1 \qquad \forall i \in I \quad (4)$$

$$\sum_{j=1}^{m} R_i^{CPU}.\ X_{ij} \le T_j^{CPU}.Y_j \ \ \forall j \in J \quad (5)$$

$$\sum_{j=1}^{m} R_i^{RAM}.\ X_{ij} \le T_j^{RAm}.Y_j \ \ \forall j \in J \quad (6)$$

$$X_{ij}, Y_j \in \{0\ ,1\} \forall j \in J\ , \forall i \in I \quad (7)$$

A major part of power consumption in a server is related to CPU, where the load is defined as a function of the CPU utilization value. Equation (1) is considered to calculate the data center load, and Equation (2) to calculate the average power consumed in the whole data center. Thus, to be the objective function of the problem examined is considered as Equation (3). The comprehensive fit function is as follows:

fitness:

$$1 - (\alpha_{RAM_i} \times \frac{RAM_i - RAM_{min}}{RAM_{max} - RAM_{min}} + \alpha_{CPU_i} \times \frac{CPU_i - CPU_{min}}{CPU_{max} - CPU_{min}}) \quad (8)$$

In this equation:

RAM_{min}: Minimum memory requested

RAM_{max}: Maximum memory requested

CPU_{min}: Minimum CPU requested

CPU_{max}: Maximum CPU requested

α_{RAM_i} and α_{CPU_i}: A random number between 0 and 1

Equation (4) shows that the i-th VM must be deployed only on one physical machine. Equation (5) shows the capacity limitation of the processor resource on the j-th physical machine, and Equation (6) the capacity limitation of the memory resource on the j-th physical machine. The latter two formulas show that the sum of resources required for VMs should be less than the capacity of physical machine resources. Equation (7) shows the range of problem variables.

Using the ABC algorithm to solve resource sharing and load balancing

FFA is used to solve this problem as follows:

First step: Receiving input data

At this stage, the data on the physical machines in the data center as well as the VM requests are received and prepared for running the algorithm.

Second step: Generation of the initial population

After receiving problem entries according to available resources and requests received, the initial population is generated; this includes a set of fireflies, each of which shows

an example of how m VMs are deployed on n physical machines. Coding the response is shown in this study. The following figure is a suggested response for deploying 5 VMs:

Vm#1	Vm#2	Vm#3	Vm#4	Vm#5
A	C	B	A	B

As the figure above shows, each response is an array equal to the number of VMs needed to be deployed on the servers. The index of each array element, the VM number, and the value corresponding to it are the server attribute intended to run the VM. For instance, server A corresponds to server B for running two VMs 1 and 4, server C running VM 2 and VM 3 and 5 in the figure above.

These early creatures are generated using GA, but as there is a resource constraint on the physical servers before production can be accepted, it must be assured of the capacity of the physical servers used, provided the conditions are met. If there is no one available, the new one will be rejected and accepted otherwise. Then the search process starts and goes through several generations, including the following steps:

Step three: using the objective function

By running the objective function for all populations, their competence and quality are specified. The objective function calculates the load consumed by all members of the population. The load consumed in each solution shows the intensity of light in that solution (bee colony). Less power consumption means more light.

Step four: Determining the best solution

The best solution for the current generation is determined according to the values already obtained. This is the best solution due to the competence function whose lowest value has the highest productivity.

Step five: Running ABC algorithm

According to the algorithm definition, worker, leader, and scout bees' movement is to find new solutions that lead to the bees location change in the search space. New solutions are created due to the movement of the bees in the search space, the search space and again have to be examined for non-violation of any capacity restrictions. The third, fourth and fifth steps continue until the highest number of generations is defined.

ABC algorithm is implemented in Matlab according to what was stated. Each evolutionary algorithm needs to initialize some basic parameters to start. These values are shown in Table 2. The following table shows the initial parameters specified in this implementation.

Table 2: Initializing parameter of the ABC algorithm

Values	Parameters
50	Population size
100	Number of iterations
50	Number of observing bees
Random	Parameter l
1	Parameter a

The duty of the ABC algorithm in the proposed approach is to determine the order of running the input requests so that the best resource sharing is obtained at the least cost. In doing so, memory and CPU parameters are used to calculate the objective function responsible for evaluating each solution to the problem. This objective function produces the same load balancing that is a number between zero and one. The closer this value is to zero, the more appropriate resource-sharing will be between requests.

ABC algorithm is shaped by using the values of Table 2 as the initialization and the following diagram shows the result of running this algorithm 100 times.

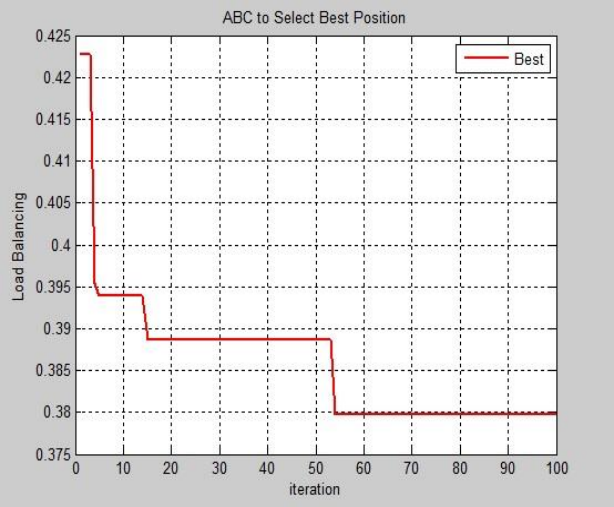

Figure 2: The results of the implementation of the ABC algorithm

The datasets used in this implementation include 50, 100, 200, 500, and 1000 (VM) requests with each request having 2 components. The first component determines the CPU requested and the second the memory requested that both requests are randomly assigned. These requests range from 0 to 100. For each physical machine (server), there are two constant values for CPU value and memory available, whose value is considered 100, which can be changed in the implementation.

Three methods used in papers [5] and [13] were used to compare the proposed method. The two methods used in these papers are GA and FFA and PSO algorithm are separate. These three algorithms are considered optimization algorithms with high efficiency in some problems. The results of the three methods are compared with the input data with the results of the proposed algorithm tested on them.

Results

The table below indicates the results of 10 times implementations of the ABC algorithm to make the obtained results more reliable. The following results are 100 for VMs and servers.

Table 3: Results of 10-time independent implementations of the program

Implementation times	Load balancing of the initial population of the ABC algorithm	Number of servers consumed out of 100 servers
1	0.42	67
2	0.45	67
3	0.48	66
4	0.55	74
5	0.41	70
6	0.32	66
7	0.52	67
8	0.55	77
9	0.41	67
10	0.44	67

Based on the results in Table (3), the results obtained are suitable for resource sharing. Values below 0.5 are considered good values for load balancing, which are usually below this value in most of the iterations. The number of servers consumed is determined by how many servers in the network each have their own memory and CPU servers to execute requests. In other words, the smaller the number of servers consumed, the better the sharing of resources and the lower the cost and time of execution of requests will be.

For a better comparison of the results, we used different numbers of VM (request) and server (resource) as shown in Figure (3).

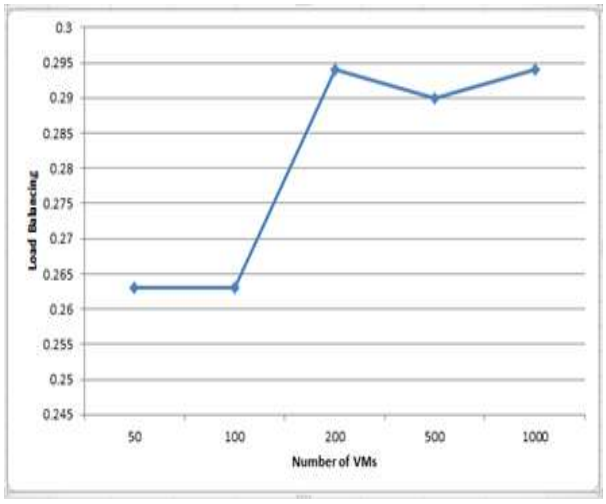

Figure 3: Comparison of the results of using different VMs

As is seen, load balancing changes in a specific range. With an increase in the number of VMs, we witness little drop in results that happens naturally with the increase in the number of machines.

Table 4: Comparison of the three implemented methods

Proposed method	GA [48]	PSO [48]	Firefly [47]	ABC
Load balancing	0.4	0.49	0.45	0.38

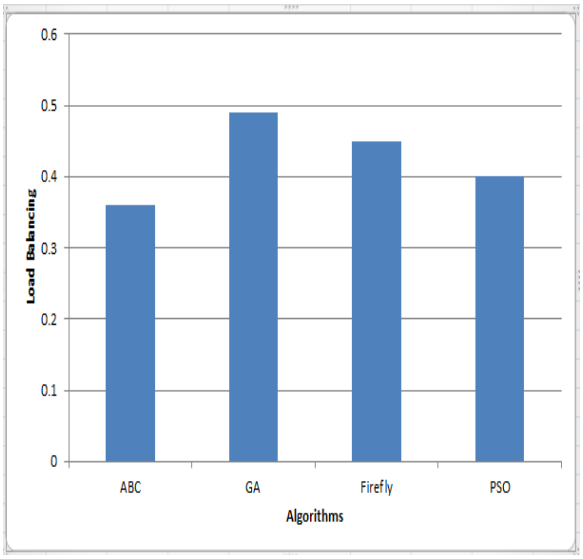

Figure 4: Comparison of various methods

As is seen, the proposed method has a better load balancing compared to the individual methods, which is due to the robustness of the ABC algorithm and the appropriate initialization in the proposed method. The ABC algorithm performs very well in discrete problems, which is seen in the results of Table 4. One of the features of evolutionary algorithms is that they depend on the problem type as well. One algorithm may give a great answer to a specific problem but may prove inadequate in another one. The ABC algorithm has shown a good response to this problem compared to other algorithms.

In the following, Makespan has been analyzed and evaluated to see the quality of service in the proposed method. Figure 5 shows this comparison:

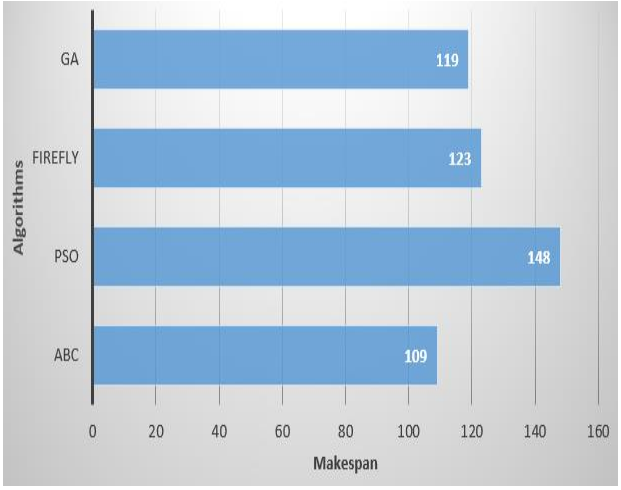

Figure 5: Comparison of the proposed method with other methods

As Figure 5 shows, the total running time in the proposed method is less than the other methods. Average running time is one of the most significant parameters in service quality and any algorithm able to reduce this value improves service quality. The ABC algorithm can provide good answers because of searching all the search space. This algorithm is suitable for discrete problems as well and can provide good results.

Conclusions

The study examined an approach for integration, resource management and power control in data centers. It was carried on by presenting a dynamic programming-based algorithm to create multiple versions of a VM without loos of performance and VM integration to minimize data center energy. A side benefit of integration is enhancing the reliability of data center services provided to customers. Using the proposed algorithm, energy savings can be achieved compared to the previous studies and using the evolutionary algorithms, one can increase energy reduction in VM location problem. Moreover, by improving these algorithms and the appropriate changes in them, one can obtain much better results. Evolutionary algorithms have great effects on optimization problems, and as in iterations the best member of the set is improved, far better results are obtained compared to other optimization algorithms.

References

1. Vaquero, L. M., Rodero-Merino, L., Caceres, J., Lindner, M. A break in the clouds: Towards a cloud definition. Sigcomm Computer Communications Review, 2009, 39(1), 30-35.
2. Banerji, R. Elastic Load Balancing in Amazon Compute Cloud. Harvard university, 2011.
3. Dasguptaa, K., Mandalb, B., Duttac, P., Jyotsna Kumar Mondald, J., Dame, S. A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing. International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA), 2013, 10, 340-347.
4. Khanchi, M., Tyagi, S. An Efficient Algorithm for Load Balancing In Cloud Computing, International Journal of Engineering Sciences & Research Technology, International Journal of Engineering Sciences & Research Technology, 2016, 5(6), 468-485.
5. Kaur, G., Kaur, K. An Adaptive Firefly Algorithm for Load Balancing in Cloud Computing. Springer, Proceedings of Sixth International Conference on Soft Computing for Problem Solving, 2017, 63-72.
6. Dhinesh Babu, L. D., Venkata Krishna, P. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing, 2013, 13(5), 2292–2303.
7. Patel, S., Patel, H., Patel, N. Dynamic Load Balancing Techniques for Improving Performance in Cloud Computing. International Journal of Computer Applications, 2016, 133-8.
8. Kao, Y. H., Krishnamachari, B., Ra, M. R., Bai, F. Hermes: Latency optimal task assignment for resource-constrained mobile computing. 2017, 16(11), 3056 – 3069.
9. Kovachev, D., Yu, T., Klamma, R. Adaptive computation offloading from mobile devices into the cloud. Proceedings of IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2012, 1, 784-791.
10. Rehman Khan, A., Othman, M., Nasir Khan, A., Akhtar Abid, Sh., Madani, S. MobiByte: an application development model for mobile cloud computing. Journal of Grid Computing, 2015, 13(4), 605-628.
11. Xu, J., Fortes, J. A. B. Multi-objective VM Placement in Virtualized Data Center Environments. IEEE/ACM International Conference on Green Computing and Communications & IEEE/ACM International Conference on Cyber, Physical and Social Computing, 2010.
12. Arianyan, E., Taheri, H., Sharifian, S. Novel energy and SLA efficient resource management heuristics for consolidation of VMs in cloud data. Computers & Electrical Engineering, 2015, 47, 222-240.
13. Shahjahan, K., Mohaimenul, K., Islam, R. Process of load balancing in cloud computing using genetic algorithm. Electrical & Computer Engineering: An International Journal (ECIJ), 2015, 4(2), 57-65.