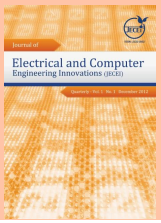
SRTTU

Journal of Electrical and Computer Engineering Innovations

JECEI, Vol. 3, No. 2, 2015

Regular Paper



A CSA Method for Assigning Client to Servers in Online Social Networks

Shahriar Minaee Jalil[1,*], Ali khaleghi[1]

[1] Imam Khomeini International University, Qazvin, Iran

*Corresponding Author's Information: minaee@eng.ikiu.ac.ir

ARTICLE INFO

ARTICLE HISTORY:
Received 24 October 2015
Revised 7 December 2015
Accepted 9 December 2015

KEYWORDS:
Online social networks
Client-server assignment
Centralized simulated
Annealing algorithm

ABSTRACT

This paper deals with the problem of user-server assignment in online social network systems. Online social network applications such as Facebook, Twitter, or Instagram are built on an infrastructure of servers that enables them to communicate with each other. A key factor that determines the facility of communication between the users and the servers is the Expected Transmission Time (ETT). A smart user-server assignment can avoid the low quality links and improve the communication between nodes and also save the valuable communication resources. Unfortunately, finding the optimal assignment turns out to be a NP-hard problem. This paper proposes the use of a heuristic algorithm named Centralized Simulated Annealing (CSA) to get a good near optimum solution for this problem. Simulation results of this investigation show that using a relatively small number of iterations, this approach achieves a very good performance improvement. On the other hand, the average number of iterations needed to achieve the near-optimal solution, will be slightly increased when the number of users in the network increase.

1. INTRODUCTION

Online social network applications rely on an infrastructure of servers that facilitate communication among their users [1]. In this scenario, users communicate to each other through the servers. Specifically, for an online social network system like Facebook or Instagram, the profile of users is saved at their primary server, and when a user (u) posts a message or any other contents in his/her network, it will be saved at his/her server (S_u). Suppose that there is a user (v) which is friend with (in connection to) u. Then, whenever v gets online, the message will be showed to v if v is located in the same server as u. Otherwise, if v is located in a different server (S_v) an extra procedure should be followed. When user v gets online, v sends a read request from S_v to S_u and S_u sends the message to S_v and v gets it from S_v. $S_u$andS_v perform the communication on behalf of their users. This architecture could be scaled up by adding servers to the network with the help of some other resources. In this indirect communication pattern, if two communicating users (friends) are located at the same server, then only one server is active to enable the communication. If the communicating users are located in two different servers, then both servers are involved in the process, which obviously maintains an increase in the communication overhead of the network.

The problem of client-server assignment in a high scale networks is crucial. The aim of this paper is to present a client-server assignment method which aims at assigning highly connected users to the same server and minimizing the Expected Transmission Tine (ETT) of users located in different servers. We assume that the messages and information that users share with each other are the mainly time-consuming

loads. We can deduce the following presumptions from the above assumption:

1- ETT for messages between users that are assigned to the same server is less than the users that are assigned to different ones. It means that if we assign all the users that communicate with each other to same server, it will reduce the total transmission time in a system.
2- Consider the case in which most of the users in the network are friends. If we assign those users to the same server to achieve a shorter transmission time, the total load of the server increases and it becomes a bottleneck. Bottlenecks often crash over time.
3- As a result of 1 and 2, the best strategy to minimize the ETT between servers is to assign the friends to the same server, and to assign users with less/no communication rate to different servers.

Above deductions indicate that a good client-server assignment depends on the communication pattern between users.

The proposed method should give a good trade-off between minimizing transmission time between severs and reducing the total ETT for servers. This method requires centralized processing of the communication pattern. Also, the complexity of obtaining the optimal solution turns out to be computationally expensive.

The main contribution of this paper is to propose an efficient way to find a good solution for this problem. This method uses the Centralized Simulated Annealing (CSA) algorithm [2]. The CSA produces optimal solutions for static systems and needs accurate information of system.

This method uses ETT between nodes as the input data to give a good client-server assignment topology.

The problem of client-server assignment can be mapped into a k-way graph partitioning problem. The k-way graph partitioning problem is defined as follows:

Consider an undirected graph $G = (V, E, c, w)$, with $n = |V|$ vertices and $m = |E|$ edges, where $c(u)$ and $w(\{u, v\})$ are non-negative weights of a vertex and an edge, respectively. Let P be a collection of k subsets $\{P_0, \dots, P_{k-1}\}$of V, we can define the set $C = \{\{u, v\} \in E | u \in P_i, v \in P_j, 0 \le i \le j \le k-1\}$ of cut edges and also the cut size

$\theta(P) = \sum_{\{u,v\} \epsilon C} w(\{u, v\})$,

which is equal to the cardinality of C if all the edges are unitary but different in all other aspects.

The problem of k-way graph partitioning with balance $1 \le \varepsilon < k$ can be described by the following model:

$$\textbf{Minimize } \theta(P)$$
$$\textbf{Subject to:}$$
$$0 < \sum_{u \epsilon P_i} c(u) \le \left\lfloor \frac{\varepsilon}{k} \sum_{v \in V} c(v) \right\rfloor$$
$$\forall i \in [0, k-1]$$
$$\bigcup_{0 \le i < k} P_i = V$$
$$P_i \cap P_j = \emptyset \quad \forall i, j \in [0, k-1] | i \ne j$$

Unfortunately, in [3] it is shown that k-way graph partitioning is a NP-hard problem, i.e., there is no solution to find the optimal answer in polynomial time. But, there are several studies focused on heuristic algorithms that tries to find a good solution for the problem. The problem of k-way graph partitioning has been studied in many fields and many heuristic algorithms have been developed for it. Some famous algorithms are Kerninghan-Lin (KL) algorithm [4] and Fiduccia-Mattheyes (FM) algorithm [5]. The authors in [6] tried to reduce the maximum flow between partitions to solve the partitioning problem. In [7-9] and [10] the proposed algorithms focus on balancing the quantity of elements in each cluster without considering the effects of edge-cut on the weights of the elements. In [11], [12] and [13], the application of k-way graph partitioning have been studied in job scheduling on distributed systems. However in, task scheduling and/or allocation problems, the total load incurred by execution tasks at processors are independent of the inter-communication load [14] [15]. In general, the k-way graph partitioning has several applicationsin problems concerning to make a balance between weight of partitions and reduce the total communication load.

The rest of this paper is organized as follows. The problem formulation and outlines our approach to tackle this problem are presented in Section 2, 3. The proposed algorithm is introduced in Section 4. Some simulation results are presented in Section 5 to show the performance of the proposed algorithm. As the final Section 6 concludes the paper.

2. PROBLEM FORMULATION

The aim of this paper is to propose a method for assigning users to servers. The assignment should be done in such a way that it reduces the ETT between users in a server as well as the transmission time between users in different servers. We will reach this goal by optimizing the objective function.

Let us introduce the variables and notations used in this paper:

- M is the number of users.
- N is the number of servers.

- G(V,E) is a weighted graph that represents communication among users. E is the set of vertices |V|=M, where each vertex represents a user. E is the set of edges that shows communication between users. An edge between two vertices V_i and V_j represents that there is a communication between these two users.
- A_{M*N} is the adjacent matrix of graph G. Any entry A_{uv}=0 indicates that there is no communication between two users such as u and v. Elements on the diagonal of the matrix will be zero because we assume that there is no self communication between users. The graph is symmetric because the cost of communication from u to v is equal to the cost of communication from v to u. We can normalize the elements of matrix by dividing any element to the sum of all elements. As a result, each element $0< A_{uv} <1$ represents a percentage of the total ETT in the network. Elements in this matrix are ETTs for transmitting a fixed size packet from u to v. The ETT index has been used to predict the expected transmission time between two nodes in a hop in large scale [16], [17].
- $X_{M*N} \epsilon \{0,1\}$: X is an assignment matrix. X_{ui}=1 if user u is assigned to server i. Since each user can be assigned to only one server, we have $\sum_{i=1}^{N} X_{ui} = 1$.
- $L_{N*N}(X)$ represents the ETT for transferring a message between two servers for an assignment matrix X. $L_{ij}(X)$ indicates the ETT for transferring a message between two different servers, and $L_{ii}(X)$ indicates the ETT for transferring a message between users inside the server i.
- $S(X)$: indicates the total ETTs for all the servers.
- Our goal is to optimize the function $S(X)$.

Minimizing $S(X)$ causes a reduction in the total ETT for transferring messages in the network. Let us assume that M, N, G and A_{M*M} are known. Our goal is to find an assignment matrix X such that it minimizes $S(X)$. Hence, the problem of assigning the clients to the servers is expressed as follow:

$$\text{Min } S(X)$$
$$\text{subject to:}$$

$$X_{ui} \epsilon \{0,1\}$$
$$\sum_i X_{ui} = 1 \quad \forall u \epsilon \{1,2,3,\dots,M\}, i \epsilon \{1,2,\dots,N\} \qquad (1)$$

For Development of the Formulation first we find the mathematical formula to calculate the matrices $L_{N*N}(X)$ and S(X) using X. At first, we find a formula to calculate the overall ETTs for transferring messages between servers. We note that this matrix is symmetric, because ETT for transferring a message from user u to v is equal to transfer a message from user v to u.

Inter-server ETTs will be calculated from ETT for transferring a message between users in any server i to another server j, assuming that $i \neq j$. We calculate this value from assignment matrix X using the following formula:

$$L_{ij}(X) = (1-\alpha) \sum_{k=1}^{M} \sum_{l=1}^{M} A_{kl} X_{ki} X_{lj} \qquad (2)$$

Internal ETTs for a server will be the ETT for transferring a message between users inside a server and it is defined as follows:

$$L_{ii}(X) = \alpha \sum_{k=1}^{M} \sum_{l=1}^{M} A_{kl} X_{ki} X_{li} \qquad (3)$$

In (2) and (3), the parameter α is a positive coefficient between 0 and 1. We know that the average ETT for transferring messages between users assigned to one server is usually much smaller than this value for users that are assigned to different servers. We use α to give more weight to the ETTs of the former compared to the latter. S(X) is the sum of all elements of L(X) for assignment matrix X and it is defined as follow:

$$S(X) = (1-\alpha)\left|\left|X^T A X\right|\right|_1 - (1 - 2*\alpha)\text{Tr}(X^T A X) \qquad (4)$$

The operator $||.||_1$ denotes the sum of all elements in a matrix. In (4), coefficient of the first expression, $(1-\alpha)$, is the weight of inter-server ETTs and coefficient of the second expression, $(1-2*\alpha)$, is the weight of internal ETTs of each server. This term is calculated as follows:

First we subtracted $(1-\alpha)$ which is the coefficient of the first term, and then we add α. If we assign all users to the same server to get a minimum total ETT, the server will be overloaded. The coefficient α regulates acceptable balance between internal ETTs and inter-server ETTs. Letting $\alpha = 0$, removes the effects of internal ETTs in the calculations. If we assume $\alpha = 1$, we have forgotten inter-server ETTs. It means that we have just minimized internal ETTs. So, the coefficient α should be considered a small value to give more weight to inter-server ETTs.

Since the problem of minimizing S(X) by changing the assignment matrix X is NP-hard, we will use a heuristic algorithm to find a good solution.

3. ALGORITHMIC OUTLINE

Here an iterative approach will be proposed. At the beginning, users are randomly assigned to the servers. At each iteration, the algorithm decides to change the assignment based on servers' information and it may move users from one server to another. Before presenting the algorithm, let us introduce the following notation. Parameter t represents iteration number. The maximum number of iterations must be known, because finding the optimal answer may take a very long time.

We add parameter t to the assignment matrix X (i.e., we have $X(t)$) to represent assignment matrix in time interval t. $S(X)$ changes to $S(X(t))$ that represents the value of goal function at time interval t. For convenience, we remove X from expressions. As an example, $S(X(t))$ changes to $S(t)$.

At each iteration of the algorithm, some users move between servers. Remind that total ETT is a part of goal function and every change in user's location will change the total ETT. Therefore, we should calculate the total ETT at each iteration when a user's location is changed. At each time interval t that user u moves from server i to server j, total ETT will be changed at servers i and j, and it remains unchanged at other servers. The changes of ETT is as follows:

The new ETT in server i is:

$$S_i(t+1) = S_i(t) - \sum_{S=1|s\neq i}^{N} \sum_{l=1}^{M} A_{ul} X_{ls}(t) \quad (5)$$

The new ETT in server j is:

$$S_j(t+1) = S_j(t) + \sum_{S=1|s\neq j}^{N} \sum_{l=1}^{M} A_{il} X_{ls}(t) \quad (6)$$

And it is un changed in other servers:

$$S_s(t+1) = S_s \quad (7)$$

As a result, the new total ETT will be defined as follow:

$$S(t+1) = S(t) + \sum_{l=1}^{M} A_{ul} \left(X_{li}(t) - X_{lj}(t) \right) \quad (8)$$

The second part in equation (5) represents the ETT of user **u** at server **i** for communicating with other servers. Index **i** will be excluded because user **u** is not in server **i** anymore. The operation reduces the value, because we should exclude ETT of user **u** from total ETT of server **i**. We repeat (6) until the operation increases the value, as a result of joining user to server **j**.

Before and after the changes in location of user, the total ETT for servers in (7) will remain unchanged except that of **i** and **j**. Equation (8) represents the summation of (5) and (6). The results are used in the algorithm.

4. ALGORITHM

In this section, we introduce the Centralized Simulated Annealing (CSA) algorithm for problem of assigning clients to servers algorithm and exploit the developed formulas of the previous section. CSA is based on simulated annealing framework and it is proved that it works well for large scale optimization problems [18] [19].

The CSA algorithm is in the class of stochastic greedy search algorithms, in wich the probability of searching in the next configuration is based on the objective value at the current and the next configuration. CSA works based on decreasing the optimization-value parameter which enables algorithm to check many configurations before it represents an optimized solution.

This stochastic search gives the ability to deal with local minimum to algorithm.

Considering goal function for the client-server assignment problem as $S(X)$, we can consider any assignment matrix X as a new configuration. There is a centralized controller that optimizes the goal function. This controller keeps communication pattern between users and servers information. The controller uses CSA to minimize goal function and find an optimized assignment matrix.

Afterwards, the controller assigns users to the server based on an optimized assignment matrix. CSA pseudo-code is displayed on Figure 1.

At the beginning, controller calculates the parameters of (8) and calculates the goal function with a randomly initialized assignment matrix. The optimization-value parameter Tb has a big value at first, enough for reaching an optimized assignment matrix. Controller creates next configuration of assignment matrix by choosing a random user and changes its location and calculates the new value of goal function (8) for this assignment matrix. If this new value is smaller than before, this assignment matrix X will be considered as a new optimized assignment matrix.

This process will continue until optimization-value parameter Tb be greater than a threshold ϵ. Each time when a new value of goal function is found, this optimization-value parameter will be reduced one degree.

Algorithm 1: Centralized SA Algorithm(CSA)

```
1.  Calculate Global parameters L, S-init
2.  Initialize Tb
3.  while Tb> ϵ do
4.     Select an user u
5.  Select  server v
6.     Calculate new-S
7.     If new-S < S-init
8.        Switch user u to server v
9.        Decrease Tb
10.    End if
11. End while
```

Figure 1: CSA algorithm pseudo-code.

5. SIMULATION RESULTS

Simulation results are represented in this section.

We initialized α with a small value (α = 0.1) to give more weight to the inter-server ETT values. This assumption is supported by the fact that really, the inter-server ETTs are usually much larger than internal ETTs at each server.

A. *Experiment 1*

At this experiment, we tested the proposed method with small made-up graphs. The ETT matrix (A_{M*N}) was filled with random values based on normal distribution. As a result of this experiment, by scaling up the users in the network, the required iterations to calculate an optimized assignment matrix will be increased. This increase is due to the increasing the number of states that a user can assign to a server with respect to the goal of problem. It is obvious that by scaling up the number of users, combination of assigning users to servers will be increased too. Figure 2 shows this argument.

By repeating this experiment 100 times for various configurations of the network, the number of users and an the average numberof iterations have been calculated.

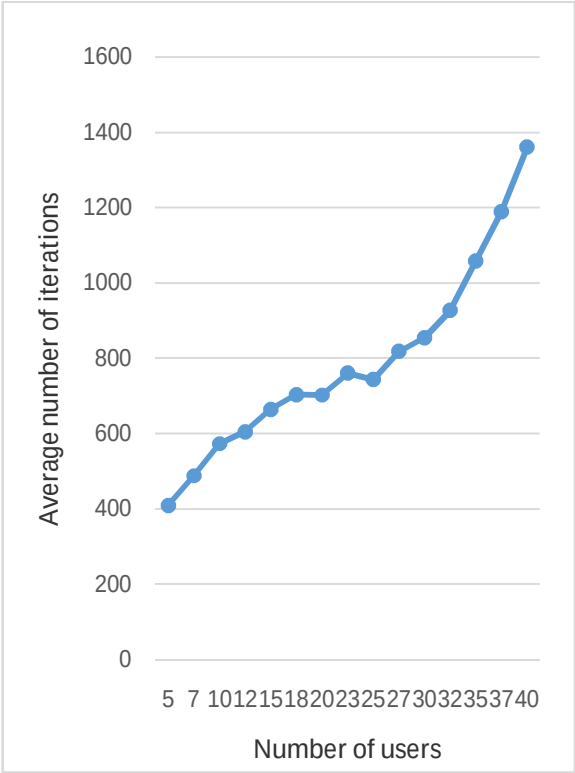


Figure 2: The average number of iterations of the algorithm in terms of the number of users.

B. *Experiment 2*

At this experiment, we calculated the average of goal function for the fixed number of iterations. We performed this experiment for two settings with 5 and 20 users. This experiment shows how the value of goal function changes through the optimization process.

In the proposed method, algorithm will continue until the value of objective function reaches to its minimum bound that represents the optimized assignment matrix. As we mentioned in Section 4, the algorithm continues its procedure until the optimization-value parameter is greater or equal to a threshold value ϵ. As indicated in Figure 3, the calculated value for objective function is at its highest level at the beginning of work.

This value will be decreased by proceeding the algorithm (increasing iterations) and gets close to its optimal value.

In Figure 3, changes in the average value of the objective function is displayed. This test was run in a system with 5 users and 4 servers in three values for optimization-value parameter Tb. We set the threshold parameter to $\epsilon = 10$. We repeated this experiment 100 times for various values of Tb. As it is displayed in Figure 3, changing Tb will change the slope of objective function. The smaller this parameter, the faster the optimized value of objective function will be found. Increasing this parameter will increase computational cost and time.

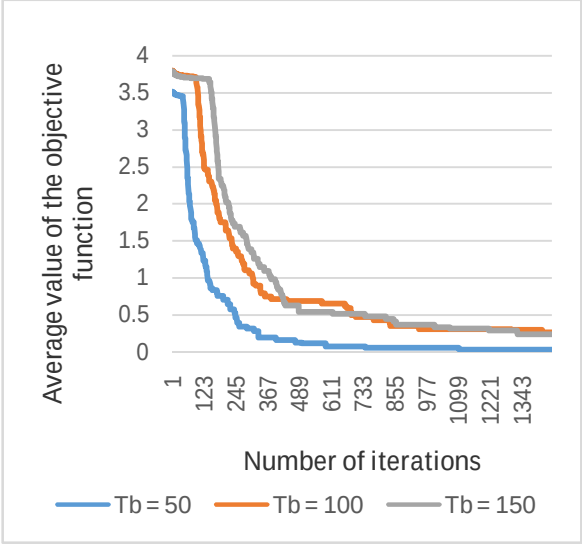


Figure 3: Changes in the average value of the objective function for the case of 5 users and ϵ=10.

We repeated experiment 2 with the same configuration, but this time we put $\epsilon = 1$. By looking at the diagram in Figure 4, it is obvious that our argument for changing Tb will change the slope of objective function and changing ϵ does not effect on it.

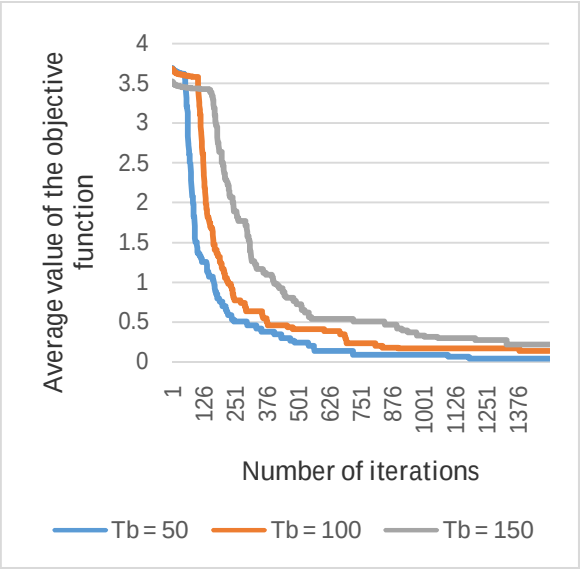


Figure 4: Changes in the average value of the objective function for the case of 5 users and ϵ=1.

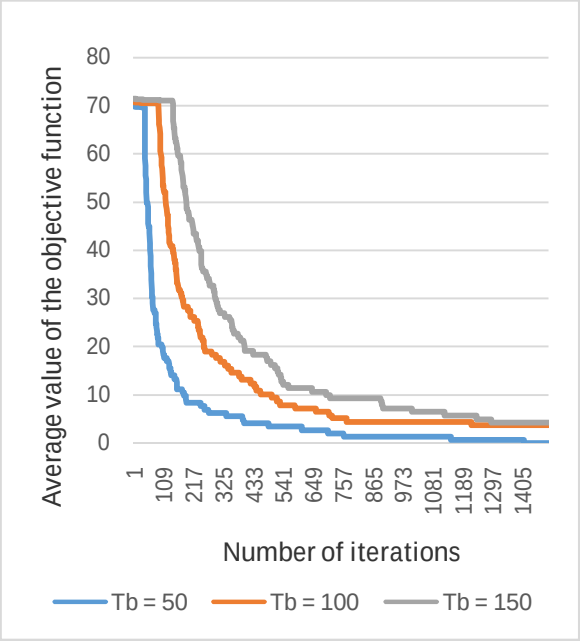


Figure 5: Changes in the average value of the objective function for the case of 20 users and ϵ=10.

In the following, we repeat the second experiment with the same configuration for 20 users. In a similar way, Figures 5 and 6 represent changes in the value of objective function in terms of the number of iterations. This time, we tested our proposed method with 20 users and for two threshold values, ϵ= 1and ϵ = 10. These two figures are the same as Figure 3 and 4 which prove our arguments. The only difference is the initial value of the objective function due to the number of users that makes the matrix of ETTs and the value of objective function larger. By proceeding the algorithm, the value of objective function becomes smaller and gets closer to its optimal value.

6. Conclusion

In this paper, the problem of assigning users to the servers in the online social network systems has been studied. Online social network applications like Facebook, Twitter or Instagram are built on an infrastructure of servers that make it possible for users of this social network to communicate with each other.

By considering the expected transmission time of messages between users, the way of assigning users to the servers is critical to reach a minimum ETT for users in a server and a balanced transmission time between servers. Our objective was to present a method for assigning the users to the servers.

A good assignment will achieve our goal. This paper has tried a metaheuristic algorithm named Centralized Simulated Annealing (CSA) to get a good near optimum solution for this problem. Simulation results showed that the algorithm searches in different configurations of assigning user to the servers until it finds the best approximately optimal solution.

Also, by scaling up the number of users, it was shown that the algorithm needs more iterations to achieve a good answer. This is because that it should check various combinations of assigning the user to the servers. In the proposed method, the algorithm starts with a randomly initialized assignment matrix. However, changing this matrix improves the value of the objective function.

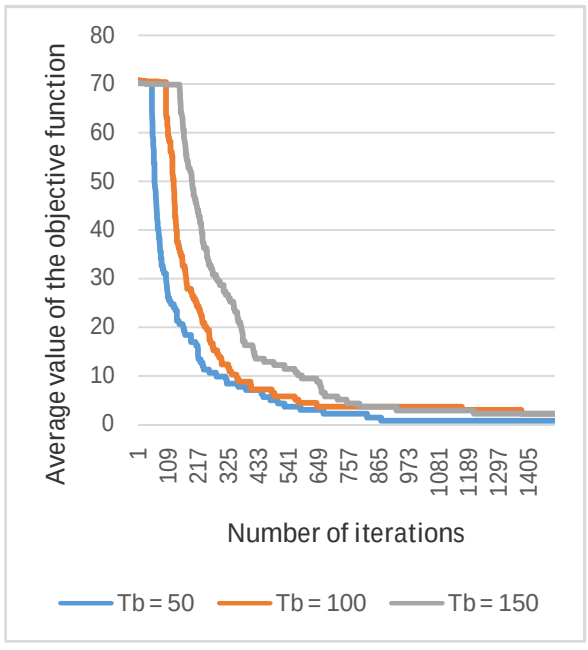


Figure 6: Changes in the average value of the objective function for the case of 20 users and ϵ=1.

REFERENCES

[1] A. Lakshman, P. Malik, "Cassandra: A decentralized structured," ACM SIGOPS Oper. Syst. Rev., vol. 44, no. [Online]. Available: http://doi.acm.org/10.1145/, pp. 35–40, Apr. 2010.

[2] K. Lee, M. El-Sharkawi, "Fundamentals of simulated annealing in modern heuristic optimization techniques: Theory and applications to power sSystems," Wiley-IEEE Press, 2008, pp. 123 - 146.

[3] R. Michael Garey, S. David Johnson, "Computers and intractability: A guide to the theory of NP-completeness," New York, NY, USA: W. H. Freeman & Co, 1979.

[4] B. W. Kernighan , S. Lin, "An efficient heuristic procedure for partitioning graphs," Bell Syst. Tech. J, vol. 49, no. 2, pp. 291–307, 1970.

[5] C. M. Fiduccia, R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in Proc. 19th Design Automation Conference, pp. 175-181, 1982.

[6] Z. Wu, R. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," IEEE Trans. Pattern Anal. Mach. Intell, vol. 15, no. 11, pp. 1101–1113, Nov. 1993.

[7] J. Shi, J. Malik, "Normalized cuts and image segmentation," *IEEE Trans*. Pattern Anal. Mach. Intell, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[8] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE Trans. Pattern Anal*. Mach. Intell, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.

[9] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis," Neural Comput, vol. 10, no. 5, pp. 1299-1319, 1998.

[10] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, "'A min-max cut," in Proc. *IEEEInt. Conf. Data Mining* (ICDM), pp. 107–114, Dec. 2001.

[11] H. S. Stone, "Multiprocessor scheduling with the aid of network flow algorithms," *IEEE Trans. Softw. Eng*, vol. SE-3, no. 1, pp. 85–93, Jan. 1977.

[12] G. Sabin, V. Sahasrabudhe, and P. Sadayappan, "On fairness in distributed job scheduling across multiple sites," *in Proc. IEEE Int. Conf. Cluster comput*, pp. 35–44, Sep. 2004.

[13] D. P. Vidyarthi, B. K. Sarker, A. K. Tripathi, and L. T. Yang, "Scheduling in distributed computing systems: Analysis, design and models," *Berlon, Germany: Springer Publishing Company*, Incorporated, 2008.

[14] Y. Jiang, Y. Zhou, and W. Wang, "Task allocation for undependable multiagent systems in social networks," *IEEE Trans. Parallel Distrib. Syst*., vol. 24, no. 8, pp. 1671–1681, Aug. 2013.

[15] Y. Jiang, J. Jiang, "Contextual resource negotiation-based task allocation and load balancing in complex software systems," *IEEE Trans. Parallel Distrib. Syst*., vol. 20, no. 5, pp. 641–653, May. 2009.

[16] D. S. J. D. Couto, "High-throughput routing for multi-hop wireless networks," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., 2004.

[17] C. A. Chen, M. Won,R. Stoleru, and G. G. Xie, "Energy-Efficient Fault-Tolerant Data Storage and Processing in Mobile Cloud," *IEEE Trans. on cloud computing*, vol. 3, no. 1, pp. 28 - 41, January. 2015.

[18] S. Kirkpatrick, M. P. Vecchi, "Optimization by simmulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[19] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Part I, graph partitioning," *Elsevier Science B.V*., vol. 37, no. 6, pp. 865–892, June. 1998.

BIOGRAPHIES

Shahriar Minaee Jalil was born in Iran in 1992. Received the B.Sc. degree in hardware engineering from the Hamedan University of Technology, Iran, in 2014, He is M.Sc. student in software engineering at Imam Khomeini International University, Qazvin, Iran from 2014.

Ali Khaleghi was born in Iran in 1973. He received the B.E. in Computer and software engineering from Azad university, Ghazvin, Iran in 1999, M.Sc. in Information Systems Management from University of Savoie, Annecy, France and Ph.D. degree in Information Systems from University of Grenoble, France in 2007.