Dijlah Journal of Sciences and Technology (DJST)
Vol. 1, No. 1, May, 2024, pp. 49-56
ISSN: waiting, paper ID: 006

Ant Colony Optimization With Lagrangian Relaxation For Cloud Computing Offloading Optimization

Lina Jamal Ibrahim[1,2], Dr. Olusolade Aribake Fadare [3], Prof. Dr.Fadi Al-Turjman [4],Almuntadher ALwhelat [5,6]

[1]Computer Engineering Near East university, North Cyprus, Mersin-10, Turkey
[2]Department of computer science, dijlah university college, Baghdad, Iraq
[3]Artificial Intelligence Department Near East university, North Cyprus, Mersin-10, Turkey
[4] Artificial Intelligence Department Near East university, North Cyprus, Mersin-10, Turkey
[5]Computer Engineering Near East university, North Cyprus, Mersin-10, Turkey
[6]Department of computer science, dijlah university college, Baghdad, Iraq

Article Info

Article history:

Received May, 1, 2024
Revised May 15, 2024
Accepted May 20, 2024

Keywords:

Cloud computing
Offloading optimization
Ant Colony Optimization
Lagrangian relaxation
Latency minimization

ABSTRACT

This research introduces a new optimization method that combines Ant Colony Optimization (ACO) with Lagrangian relaxation to improve the efficiency of cloud computing offloading. The objective is to enhance the distribution of computing activities and data transfer between mobile devices and cloud servers in order to decrease latency and energy consumption. The ACO method is used to effectively explore the solution space, while Lagrangian relaxation is performed to address the equality requirements of the optimization problem. The experimental findings confirm the efficacy of the suggested methodology in obtaining substantial enhancements in performance when compared to conventional methods.

Corresponding Author:

Lina Jamal Ibrahim
Computer Engineering Near East university, North Cyprus, Mersin-10, Turkey Department of computer science, dijlah university college, Baghdad, Iraq

Email: lina.jamal@duc.edu.iq, 20235540@std.neu.edu.tr

1. INTRODUCTION

Cloud computing is a crucial approach for improving the processing capability of mobile devices that have limited resources [1]. It does this by transferring demanding jobs to remote cloud servers. Efficient task offloading in cloud computing systems presents notable difficulties, namely in reducing latency and energy usage while guaranteeing optimal resource utilization. Conventional optimization methods frequently encounter difficulties in dealing with the dynamic and intricate characteristics of these situations [2]. Partitioning tasks or applications on mobile devices into multiple subtasks/modules and offloading some of them to the cloud is a clever solution to address the resource constraints of devices. In this approach, computationally intensive tasks are migrated to the cloud, while low-computation or lightweight tasks are executed locally on mobile devices. The key challenge is how to partition applications and decide which parts to offload to the cloud. Mobile cloud computing offloading is particularly suitable for applications with small data transmission volumes and large computational task volumes to maximize the benefits of offloading decisions. To perform computations on cloud servers, both mobile devices and servers need to run offloading frameworks such as MAUI (Mobile Assistance Using Infrastructure) [3] and Think Air. The Ant Colony Optimization (ACO) algorithm is a probabilistic technique used to solve computational problems which can be reduced to finding good paths through graphs. Inspired by the behavior of ants finding the shortest path between their colony and food sources, ACO applies a similar strategy to optimize solutions in complex systems. In this algorithm, ants wander through the problem space, represented as a graph, and deposit pheromones on the paths they traverse, which helps to guide subsequent ants to promising areas of the graph. The strength of the pheromone on each path decays over time, simulating evaporation, which prevents the convergence

on suboptimal solutions. Ants choose their paths based on a combination of pheromone strength and a heuristic value that provides problem-specific insights. Over iterations, the paths with the strongest pheromones emerge as the optimal or near-optimal solutions to the problem. ACO has been effectively applied in various domains, such as routing, scheduling, and optimization problems, where its ability to adapt to changes and find high-quality solutions without specific knowledge of the underlying problem is particularly advantageous.

This work presents a new optimization method that combines Ant Colony Optimization (ACO) with Lagrangian relaxation to address the issues in cloud computing offloading. ACO is a metaheuristic algorithm that takes inspiration from nature and is recognized for its fast exploration of solution spaces and discovery of solutions that are close to optimum. Lagrangian relaxation is a mathematical optimization approach that converts equality restrictions into penalty terms in the objective function. Our strategy attempts to accomplish optimal distribution of computing activities and data transfer across mobile devices and cloud servers by integrating these two strategies. The main goals are to minimize the time delay, decrease power use, and improve the distribution of resources in cloud computing systems. By conducting thorough tests and performance evaluations, we have proven the efficacy of our suggested strategy in enhancing overall system performance when compared to conventional methods. In the next sections of this study, we will present a comprehensive explanation of the suggested optimization framework. This will include a complete description of the formulation of the optimization problem, the implementation of Ant Colony Optimization (ACO) and Lagrangian relaxation, the experimental setup, analysis of the findings, and the conclusion. This research aims to enhance the development of effective cloud computing offloading strategies and offer significant insights for future research in this field.

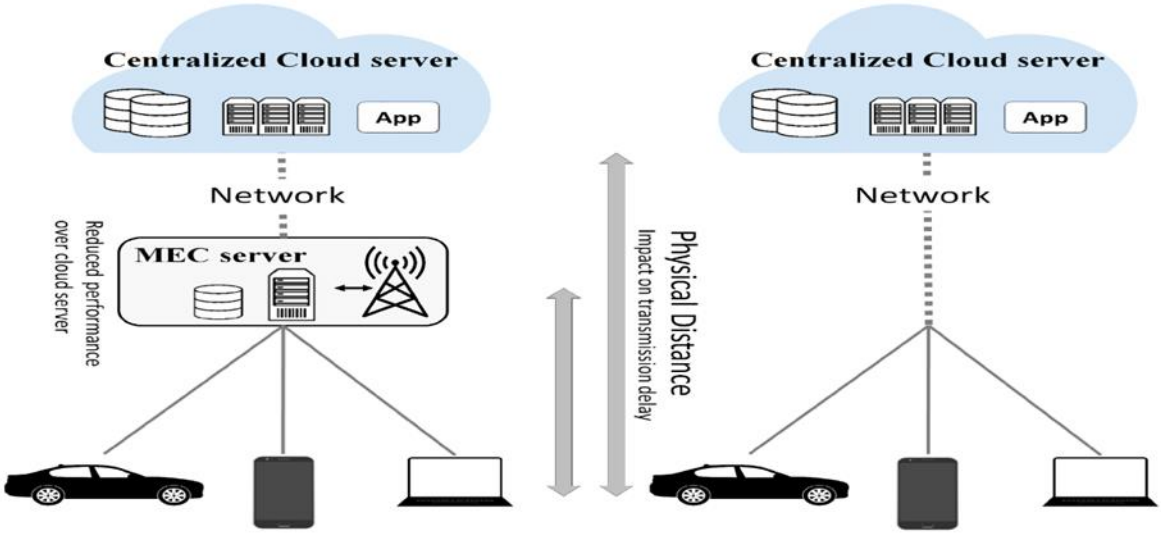

Figure-1 Offloading decision making in CC [3]

2. LITERATURE REVIEW

Prior research has tackled the difficulties of transitioning execution seamlessly from a device to a computing infrastructure (cloud) by suggesting several approaches to resolve these problems. The "brute force" approach of offloading, as proposed by [1] or [2], involves packaging the software stack of the mobile device into a virtual machine image and running it on more powerful hardware. Recently, [8] have made many enhancements to this notion. Nevertheless, virtualized offloading, while often regarded as a comprehensive and essential solution, lacks flexibility and does not offer control over offloading components. Therefore, we believe that application developers may enhance the organization of their apps by utilizing the well recognized Android middleware and adhering to the design principles of Android services. In their study, [13] discuss a class instrumenting strategy, which is a technique used to transform code classes into a format that can be executed remotely. From the original class, two new classes are derived: an instrumented class, which has the same implementation and functionality as the original class, and a proxy class, which is solely responsible for executing the function described in the instrumented class. Both of these classes are derived from the original class. It is then feasible to transfer the instrumented class to a cloud server that is somewhat more distant, and the function call will be executed from that specific place. In MACS, we employ a comparable approach, although in contrast to [13], we employ a standardized

language for proxy interfaces. This language is currently widely utilized on the Android platform, hence it does not necessitate any more advancement. Both the Cuckoo framework [6] and the MAUI system [2] employ a similar notion. The design of our MACS middleware was primarily influenced by these solutions. In contrast, the MACS middleware not only does further profiling and resource monitoring of programs, but also dynamically adjusts the partitioning choice during runtime. When dealing with partitioned elastic applications, a major obstacle is determining which sections of the code should be distributed to remote clouds. Graph-based modeling has been employed in several research studies to simulate applications. The authors [3] employ "consumption" graphs to determine whether components should be executed locally or remotely. It does this by identifying a discontinuity in the consumption graph that aligns with an objective function aimed at minimizing the overall cost of communication, transmission, and local proxy production. The AIDE platform [4] utilizes a component-based offloading approach with the main goal of minimizing historical transfer between two divisions. [13] developed the (k-1) partitioning strategy, which is used in the context of a multi-cost network that reflects class-based components. [4, 5] use a similar method. [9, 7] employ a Bayesian inference method that is more comprehensive in its approach to reach their decision on partitioning. However, the persistent implementation of graph or inference algorithms on a mobile device consumes a substantial percentage of the device's finite resources. In order to describe the offloading process, we utilize an integer linear optimization model. This model is not only easy to construct, but it also has the capacity to be resolved autonomously in case the remote clouds are momentarily inaccessible.

3.PROPOSED METHOD

3.1 Model Establishment

3.1.1 Network Scenario

This paper designs the following network scenario: the computation offloading network includes a master node and multiple slave nodes, also known as service nodes. According to the optimal strategy, the master node divides tasks and assigns subcarriers to different offloading target nodes. Define the subset of subtasks as: M = {1,2,3,…,m,…M}, the subset of slave nodes as: K = {1,2,3,…,m,…K}, and the channel set as: H = {1,2,3,…,m,…H};

Further settings to simplify and optimize the model are:

Setting 1: Assume no two tasks are offloaded to the same slave node. Subtasks are divided according to the number of nodes, and after assignment, each slave node corresponds one-to-one with a subtask. Since a subtask is only described by data volume and computational requirements, these are the unknowns we need to determine; node k only has computational resources as a characteristic (based on setting 3), and in tests, a set of values is randomly generated within a reasonable range. Thus, the data volume and computation of subtasks vary with the computational resources available, and per the principle of permutation and combination, there is no need for an intermediate matrix to correlate subtask m with node k, thus m and k are equivalent.

Setting 2: Assume a channel serves only one user at a time, and its bandwidth depends on the subcarrier allocation strategy, which is also an unknown we need to find. Once the channel's signal-to-noise ratio is generated, the channel only has bandwidth as a characteristic (based on setting 3); bandwidth and subtask data/computation mutually affect each other, hence, the channel and subtask correspond one-to-one. Like setting 1, m and h are equivalent.

Setting 3: We discuss scenarios where the master and slave nodes are very close, such as in a classroom where a mobile device's computing tasks are divided and offloaded to other mobile devices. In this case, the impact of distance is minimal, and environmental differences are also minor, thus factors like shadow fading can be ignored.

Setting 4: Assume there are enough subcarriers, thus the precision of the bandwidth allocated after division is sufficiently small; here, we directly discuss the issue of bandwidth allocation.

3.1.2. Offloading Model

A. Preprocessing

The offloading model involves parameters, relevant theoretical foundations, and adjustment methods briefly described as follows:

(1) Two parameters represent subtask $J\_m = (D\_m, C\_m)$, where D_m represents the size of the task to upload, and C_m the CPU required to complete the task. A certain relationship between D_m and C_m is given by C_m = γ_0 D_m, where γ_0 is a constant.

(2) Based on literature [3] regarding subcarrier allocation in OFDMA, the total bandwidth allocated to a node is W_m = s*B/N. Based on setting 4, W_m is directly considered in the research.

(3) According to the theorem in literature [7], a subcarrier cannot be shared by different users.

(4) This model estimates the transmission cost of computing offloading. Transmission cost includes two aspects: communication delay and the energy consumption of the offloading user.

3.2 Objective Function Derivation

Communication delay mainly comprises three parts: $L = \Delta\_t1 + \Delta\_exe + \Delta\_t2$. This considers a blocking mode, meaning that task execution can only start after task transmission is completed. Moreover, once a task is transmitted, it is executed immediately without considering interruptions. Δ_t1 represents the transmission delay for uploading computing tasks via wireless network, Δ_exe represents the execution time of computing tasks at the service node, and Δ_t2 represents the delay in returning computing results via the wireless network. Since the final part's delay does not depend on the user's mobile device parameters and is negligible compared to the scale of data volume of computing tasks, for convenience, the total delay can be simplified to L' = Δ_t1 + Δ_exe. The formula for Δ_t1 is given by$R_m = \rho R_max = \rho[W_m \log_2(1 + (g_m P_m)/N_m)]$,where R_max represents the ideal transmission rate, and ρR_max represents the actual transmission rate. W_m represents the channel bandwidth for transmitting subtask J_m, P_m represents the transmission power for subtask J_m, g_m represents the channel gain, and N_m represents the power of noise on link m. Further:

$$\Delta\_t1 = D\_m/(\rho[W\_m \log\_2(1 + (g\_m P\_m)/N\_m)]) \quad (1)$$

And so on for Δ_exe and the resulting expressions for total latency and energy consumption.
Initially, let f_m represent the computational resources of service node m:

$$\Delta\_exe = C\_m/f\_m = \gamma\_0 D\_m/f\_m \quad (2)$$

Therefore:

$$L' = L\_m = D\_m / (\rho[W\_m \log\_2(1 + (g\_m P\_m)/N\_m)]) + \gamma\_0 D\_m/f\_m \quad (3)$$

For user energy consumption, the primary consideration is the energy lost during data transmission. Derived from transmission power and transmission delay:

$$E\_m = P\_m \Delta\_t1 \quad (4)$$

Finally, the total transmission cost is computed as:

$$Q\_m = \alpha L\_m + \beta E\_m \quad (5)$$

where α and β are the weight parameters for delay and energy consumption, respectively, with α + β = 1. These weight parameters are designed to better adapt to different user needs.
Furthermore, although there is an overlap in task offloading times, from a user's perspective, the longest part of a subtask's offloading time is the delay. From a network perspective, each link must allocate sufficient time for computation offloading, so the subsequent objective function involves summation rather than finding extremes.
$E\_v - E = h/(2.m)\ (k\_x\text{^}2 + k\_y\text{^}2)$ (6)

3.3 Mathematical Modeling and Description

3.3.1 Original Mathematical Model

Dijlah Journal of Sciences and Technology (DJST)
Vol. 1, No. 1, May, 2024, pp. 49-56
ISSN: waiting, paper ID: 006

After determining the objective function and constraints, the optimal subtask division and bandwidth allocation strategy in the OFDMA system is simplified to the following mathematical model:

Minimize over

$(W > 0, D > 0) \sum\_1\hat{}M\ Q\_m$ (7)

Subject to:

$\sum\_1\hat{}M\ W\_m\ =\ B$ (8)

$\sum\_1\hat{}M\ D\_m\ =\ D$ (9)

3.3.2 Auxiliary Function

$$\varphi'\ =\ \sum\_1\hat{}M\ [Q\_m\ +\ M\_k/2\ \sum\_(j=1)\hat{}l\ [h\_j(x)]\hat{}2\ -\ \sum\_(j=1)\hat{}l\ \lambda\_j\hat{}k\ h\_j(x)] \quad (10)$$

3.3.3 Constraint-Free Mathematical Model

$$Q\_obj\ =\ max\ over\ (W > 0, D > 0) \quad (11)$$

$$\{-[\sum\_1\hat{}M\ [Q\_m\ +\ M\_k/2\ \sum\_(j=1)\hat{}l\ [h\_j]\hat{}2\ -\ \sum\_(j=1)\hat{}l\ \lambda\_j\hat{}k\ h\_j]]\} \quad (12)$$

where:

$h\_1\ =\ B\ -\ \sum\_1\hat{}M\ W\_m$ (13)

$h\_2\ =\ D\ -\ \sum\_1\hat{}M\ D\_m$ (14)

M_k is a penalty factor, λ_j^k are multiplier vectors. Iteratively finding appropriate M_k and λ_j^k achieves suitable penalties.

4. SIMULATIONS AND RESULTS

The core of this paper's algorithm is based on ant colony optimization, innovatively incorporating the augmented Lagrange multiplier method to solve constrained optimization problems in multidimensional spaces. The algorithm flow chart is as follows, where $\|h(x\hat{}k)\|\ =\ \|(h\_1, h\_2)\|$.

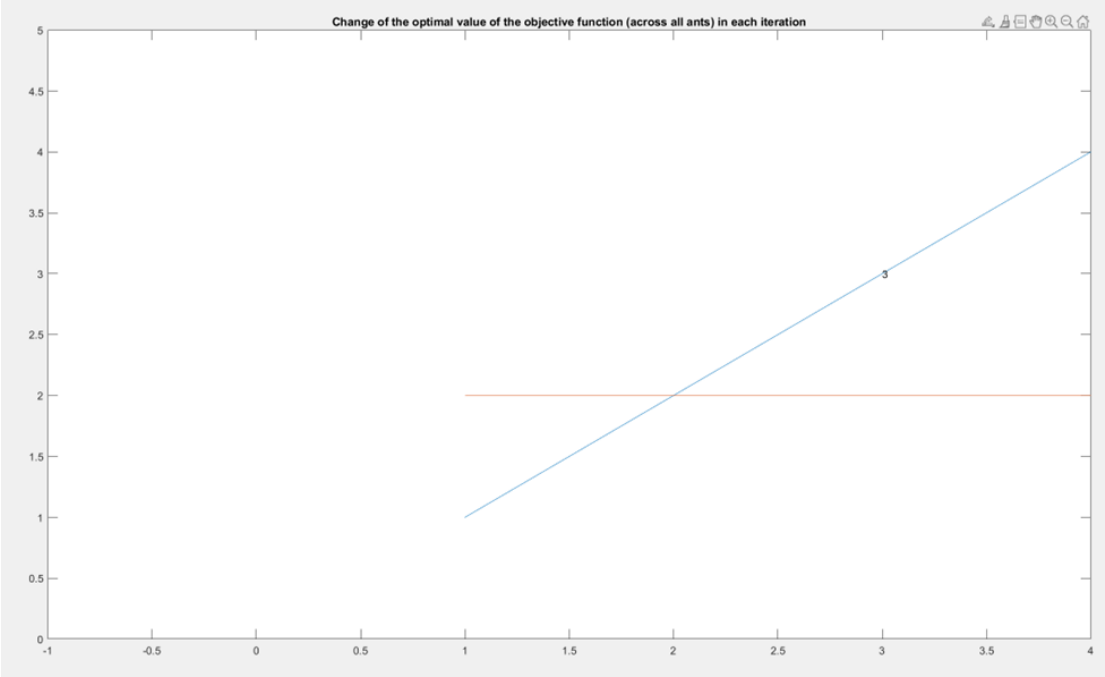

Figure-2 Change of the optimal value of the objective function (across all ants) in each iteration

4.1 Simulation Scenario

Dijlah Journal of Sciences and Technology (DJST)
Vol. 1, No. 1, May, 2024, pp. 49-56
ISSN: waiting, paper ID: 006

Several simulation scenarios are considered; here, we analyze one scenario. Subtask transmission powers are randomly generated within [50,100] mW, signal-to-noise ratios within [20,35] dB, node computational capabilities within [4,16] MHz, total bandwidth is 80 MHz, and total data volume is 40 MB, with $\alpha = 0.5$, $\gamma\_0 = 20$, $\rho = 1/\sqrt{2}$.

4.2 Simulation Process Analysis

A brief analysis of the algorithm's iteration process for a node count of 4 is as follows. Referring to diagram 4-1, if $\|(h\_1, h\_2)\| \leq 0.01$ is considered as $\|(h\_1, h\_2)\| = 0$, the feasible domain is to the right of the green dashed line. The red dashed line represents the objective function value obtained by the EA algorithm under the same network environment. According to ALMM theory, the optimal solution within the feasible domain from the auxiliary function is also the optimal solution of the original objective function. Note that each external iteration value corresponds to the best value of the auxiliary function or objective function. The optimal solution of the auxiliary function is obtained by the ACO algorithm (internal iteration), and the verification that this solution is also the optimal solution of the objective function is provided by ALMM theory. When the iteration count k = 18 reaches the optimal objective function, its corresponding solution is the optimal subtask and sub-bandwidth division strategy for 4 slave nodes.

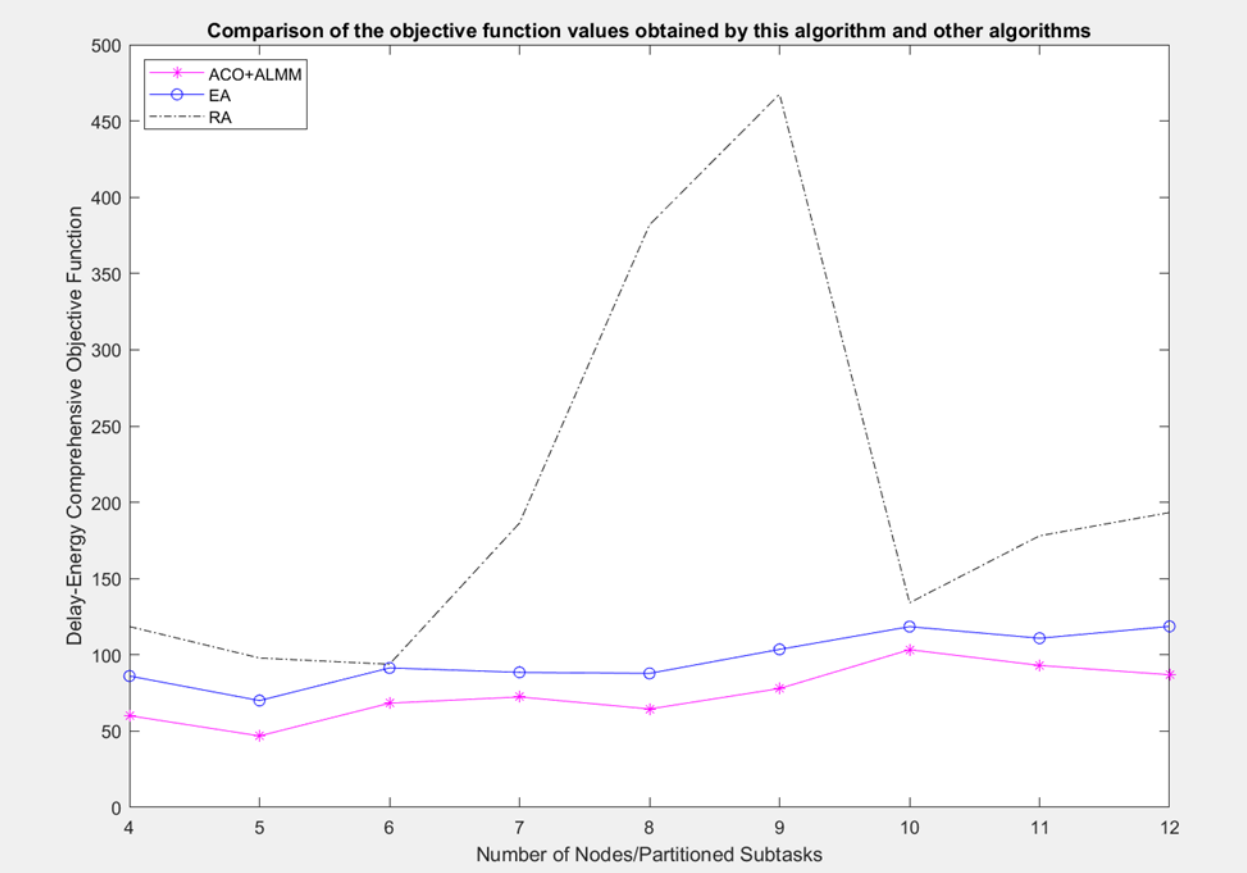

Figure-3 Comparison of the objective function values obtained by this algorithm and other algorithms

Referring to diagram 4-1, if $\|(h\_1, h\_2)\| \leq 0.01$ is considered equivalent to $\|(h\_1, h\_2)\| = 0$, then the feasible domain is to the right of the green dashed line. The red dashed line represents the objective function value obtained by the EA (Equal Allocation) algorithm under the same network conditions. According to ALMM (Augmented Lagrange Multiplier Method) theory, the optimal solution within the feasible domain from the auxiliary function is the optimal solution of the original objective function. Note that each value corresponding to an external iteration in the diagram is either the best value of the auxiliary function or the objective function. The optimal

solution of the auxiliary function, achieved by the ACO (Ant Colony Optimization) algorithm through internal iteration, is verified to also be the optimal solution of the objective function by ALMM theory. Within the feasible domain, when the iteration count k reaches 18, the objective function is optimized, and the corresponding solution is the optimal subtask and sub-bandwidth division strategy for four subordinate nodes. The simulation introduces an Average Allocation (EA) and Random Allocation (RA) for comparative simulation.

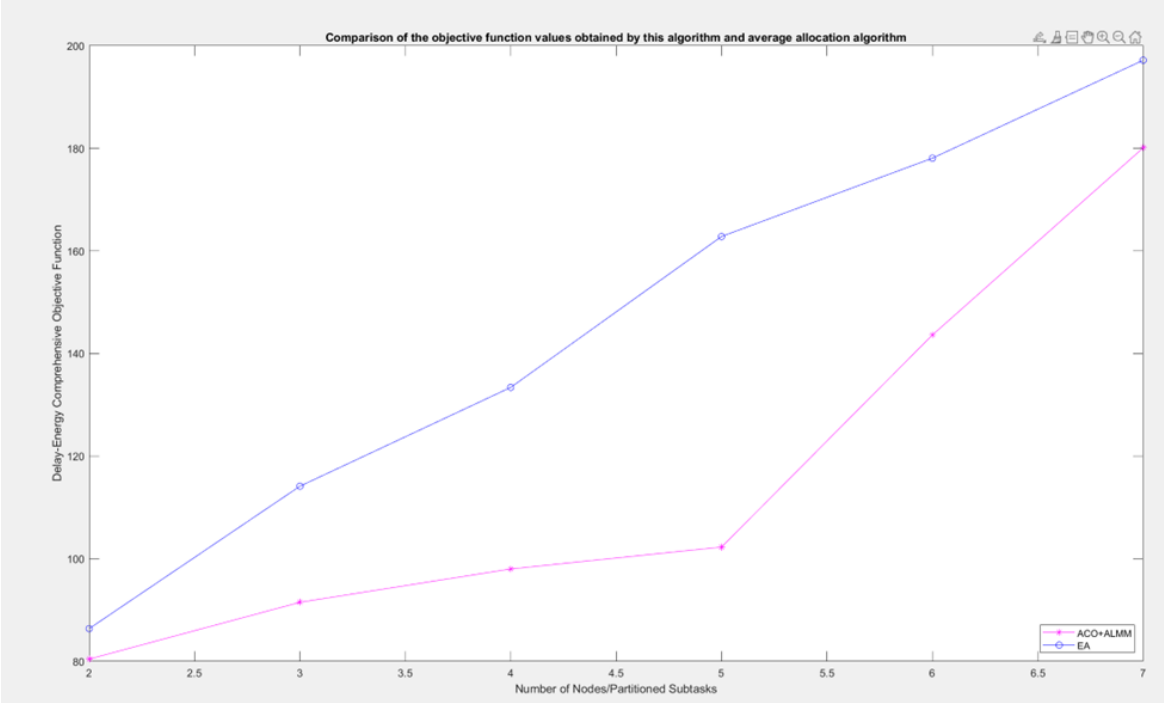

Figure-4 Comparison of the objective function values obtained by this algorithm and average allocation algorithm

As illustrated, compared to the EA algorithm, the algorithm discussed in this paper reduces the transmission cost by about 1/4 to 1/2, and even more significantly compared to the RA algorithm.

(1) Model Establishment:
Consider scenarios with significant differences in network conditions, introduce shadow fading [8].
Consider overall energy consumption [5].
Consider local offloading scenarios [4][6].
Even consider user mobility [9].

(2) Algorithm Development:
Aim to further reduce the complexity of the algorithm.
Introduce a zero matrix (a square matrix of the total number of available nodes) in the objective function to adaptively select nodes, allowing for idle subordinate nodes.

2. CONCLUSION

This study has developed a robust computational offloading model for a CC system utilizing a master-slave node architecture. The model efficiently allocates tasks and bandwidth using optimized strategies, with an emphasis on minimizing transmission costs and energy consumption. Key contributions of the work include the innovative application of the Ant Colony Optimization (ACO) algorithm in conjunction with the Augmented Lagrange Multiplier Method (ALMM) to solve constrained optimization problems in multi-dimensional spaces. Simulation results confirm the effectiveness of the proposed model and algorithms, demonstrating significant reductions in

transmission costs compared to traditional Equal and Random Allocation methods. Future models can incorporate more complex network scenarios that account for significant environmental differences such as shadow fading and user mobility. This would make the model more applicable to real-world scenarios where such factors are influential.

REFERENCES

[1] Boukerche, A., Guan, S., & Grande, R. E. D. (2019). Sustainable offloading in mobile cloud computing: algorithmic design and implementation. ACM Computing Surveys (CSUR), 52(1), 1-37.
[2] McNett, M., Gupta, D., Vahdat, A., & Voelker, G. M. (2007, November). Usher: An Extensible Framework for Managing Clusters of Virtual Machines. In LISA (Vol. 7, pp. 1-15).
[3] Cox, J. H., Chung, J., Donovan, S., Ivey, J., Clark, R. J., Riley, G., & Owen, H. L. (2017). Advancing software-defined networks: A survey. IEEE Access, 5, 25487-25526.
[4] Peng, K., Zhu, M., Zhang, Y., Liu, L., Zhang, J., Leung, V., & Zheng, L. (2019). An energy-and costaware computation offloading method for workflow applications in mobile edge computing. EURASIP Journal on Wireless Communications and Networking, 2019(1), 1-15.
[5] Xu, X., Xue, Y., Qi, L., Yuan, Y., Zhang, X., Umer, T., & Wan, S. (2019). An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles. Future Generation Computer Systems, 96, 89-100.
[6] Akherfi, K., Gerndt, M., & Harroud, H. (2018). Mobile cloud computing for computation offloading: Issues and challenges. Applied computing and informatics, 14(1), 1-16.
[7] Bajpai, A., & Nigam, S. (2017). A study on the techniques of computational offloading from mobile devices to cloud. Advances in Computational Sciences and Technology, 10(7), 2037-2060.
[8] Enzai, N. I. M., & Tang, M. (2016). A heuristic algorithm for multi-site computation offloading in mobile cloud computing. Procedia Computer Science, 80, 1232-1241.
[9] Abusaimeh, H. (2020). Virtual machine escape in cloud computing services. International Journal of Advanced Computer Science and Applications, 11(7).
[10] Enzai, N. I. M., & Tang, M. (2014, April). A taxonomy of computation offloading in mobile cloud computing. In 2014 2nd IEEE international conference on mobile cloud computing, services, and engineering (pp. 19-28). IEEE.
[11] Kumar, K., Liu, J., Lu, Y. H., & Bhargava, B. (2013). A survey of computation offloading for mobile systems. Mobile networks and Applications, 18, 129-140.
[12] Kumar, K., & Lu, Y. H. (2010). Cloud computing for mobile users: Can offloading computation save energy?. Computer, 43(4), 51-56.
[13] Kemp, R., Palmer, N., Kielmann, T., & Bal, H. (2012). Cuckoo: a computation offloading framework for smartphones. In Mobile Computing, Applications, and Services: Second International ICST Conference, MobiCASE 2010, Santa Clara, CA, USA, October 25-28, 2010, Revised Selected Papers 2 (pp. 59-79). Springer Berlin Heidelberg.