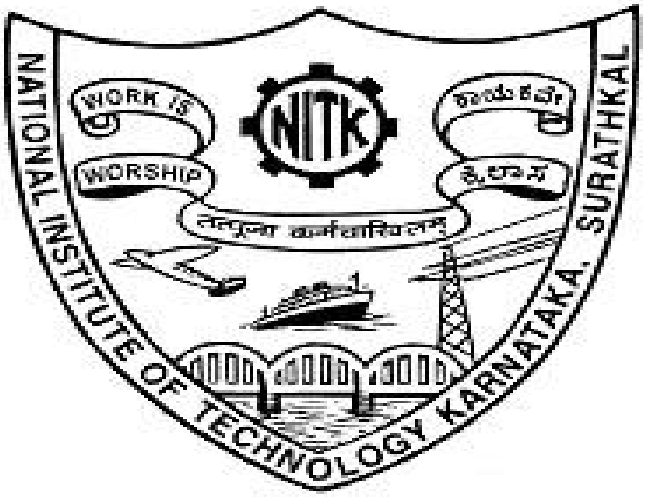
# CLOUD COMPUTING ASSIGNMENT

# REPORT

## *Energy and Delay Minimization of Partial Computing Offloading for D2D-Assisted MEC Systems*




**Team Members,**

Mahadev Hatti - (191CS133)
Vithal Mhetre - (191CS262)
Shrinivas Bagadi - (191CS156)
Yash Sharma - (191CS164)

**Submitted To,**

Dr. Sourav K. Addya
Asst Professor
CSE Department
NITK Surathkal

➔ Github Link to the Project 🔗

# INTRODUCTION

The rapid development of mobile devices has created a huge demand for mobile computing. As a result, mobile edge computing (MEC) has emerged as a promising technology to offload computational tasks from mobile devices to nearby servers, improving the performance and reducing energy consumption. However, offloading data to the cloud or remote servers may result in high latency and limited bandwidth, which may not be suitable for delay-sensitive applications.

To address this issue, device-to-device (D2D) communication has been introduced to enable direct data exchange between nearby mobile devices. Partial computing offloading (PCO) is a novel concept that utilizes D2D-assisted MEC systems to minimize energy consumption and reduce the delay in mobile computing. In this context, Haipeng Wang, Zhipeng Lin, and Tiejun Lv have conducted research on the energy and delay minimization of PCO for D2D-assisted MEC systems.

Their research paper titled "Energy and Delay Minimization of Partial Computing Offloading for D2D-Assisted MEC Systems" provides an in-depth analysis of the PCO technique and its application in D2D-assisted MEC systems. The authors propose an optimization model for energy-efficient and delay-aware PCO, taking into account the characteristics of mobile devices and the network environment.

One of the key contributions of the paper is the development of two algorithms for PCO in D2D-assisted MEC systems. Algorithm 1 is a greedy algorithm that selects the optimal set of devices to offload the computation tasks based on the device's energy consumption and processing capability. Algorithm 2 is a dynamic programming algorithm that optimizes the energy consumption and delay performance of PCO by considering the interference between devices and the network environment.

The purpose of this report is to provide an overview of the research conducted by Wang et al. and to present a critical evaluation of their findings. This report aims to explore the significance of their research in the field of mobile edge computing, and to provide a comprehensive understanding of the PCO technique for D2D-assisted MEC systems. Additionally, this report will provide an in-depth analysis of Algorithm 1 and Algorithm 2, highlighting their strengths and weaknesses, and discussing their practical implications.

## WorkFlow

**Algorithm 1 The Knapsack Problem-based Pre-Allocation (PA) Algorithm**

Input:$V_L$, $V_K$, pmax, $\psi$, D, C

Output:M, K ,L ,$\gamma$

1: Initialization: F[0][]←{0}

2: Initialization:F[][0]←{0}

3: for i←1 to N do

4: for k←1 to $V_L$ do

5: F[i][k]←F[i−1][k]

6: if(k≥$D_iC_i$)

7: then F[i][k] ← max(F[i][k], F[i−1][k−$D_iC_i$]+$D_iC_i$)

8: end for

9: end for

10: i←N

11: j←$V_L$

12: while(i>0 && j>0)do

13: if(F[i][j]=F[i−1][j−$D_iC_i$]+$D_iC_i$) then

14: Li ← $D_iC_i$

15: j←j−$D_iC_i$

16: end if

17: i←i−1

18: end while

19: same way obtain K

20: M=$\psi$−K−L

The pre-allocation step in the PA Algorithm is the key contribution of this algorithm. By pre-allocating the computation tasks based on the Knapsack Problem, the algorithm can effectively optimize the partial offloading decision and reduce the energy consumption and delay in D2D-assisted MEC systems

The PA Algorithm consists of the following steps:

1. Initialization: Initialize the system parameters, including the computation and communication resources of the device and the edge server.
2. Task partitioning: Partition the computation tasks into two parts: one part is offloaded to the edge server, while the other part is computed locally on the device.
3. Task pre-allocation: Use the Knapsack Problem to pre-allocate the computation tasks between the device and the edge server based on the weights and values of the tasks and the resource constraints of the devices and edge servers.
4. Communication allocation: Determine the optimal allocation of the communication resources between the device and the edge server to minimize the energy consumption and delay.
5. Execution: Execute the computation tasks based on the allocation results.

## Algorithm 2 Alternate optimization Algorithm

Input: $V_{new} = \max\{T^{L}_{sum}, T^{D2D}_{max}, T^{MEC}_{max}\} + k_1 E^{tra}_{sum} + k_2 E^{ex}_{esum}$,

pbegin = $p_{max}$, $I_{max} = 50$ ,$u = 10^{-5}$, $V_{new} + 10 = V_{old}$, $F_{mec}$, $F_{ue}$

Output: $\gamma$, $L^*$ ,$K^*$ ,$M^*$ ,$P$

1: while $V - V_{new} \geq u$ and $I \leq I_{max}$

2: $I = I+1$

3: $V_{old} = V_{new}$

4: update $\gamma$, L*, K*, M * using algorithm 1

5: update P by solving $P_2$

6: $V_{new} = \max\{T^{L}_{sum}, T^{D2D}_{max}, T^{MEC}_{max}\}+k1E^{tra}_{sum}+k2E^{exe}_{sum}$

7: end while

1. The algorithm takes inputs of various time and energy parameters, along with a constant k1 and k2.
2. The output of the algorithm is a set of values that minimize the objective function V, which is defined as the sum of various time and energy parameters.
3. The algorithm uses a while loop to iteratively update the values of $\gamma$, L*, K*, and M* using Algorithm 1.
4. The algorithm then updates the value of P by solving P2.
5. After each iteration, the algorithm calculates a new value of V and compares it to the previous value of V. If the difference is less than a given threshold u, the algorithm terminates.
6. The algorithm returns the values of $\gamma$, L*, K*, M*, and P that minimize the objective function V.

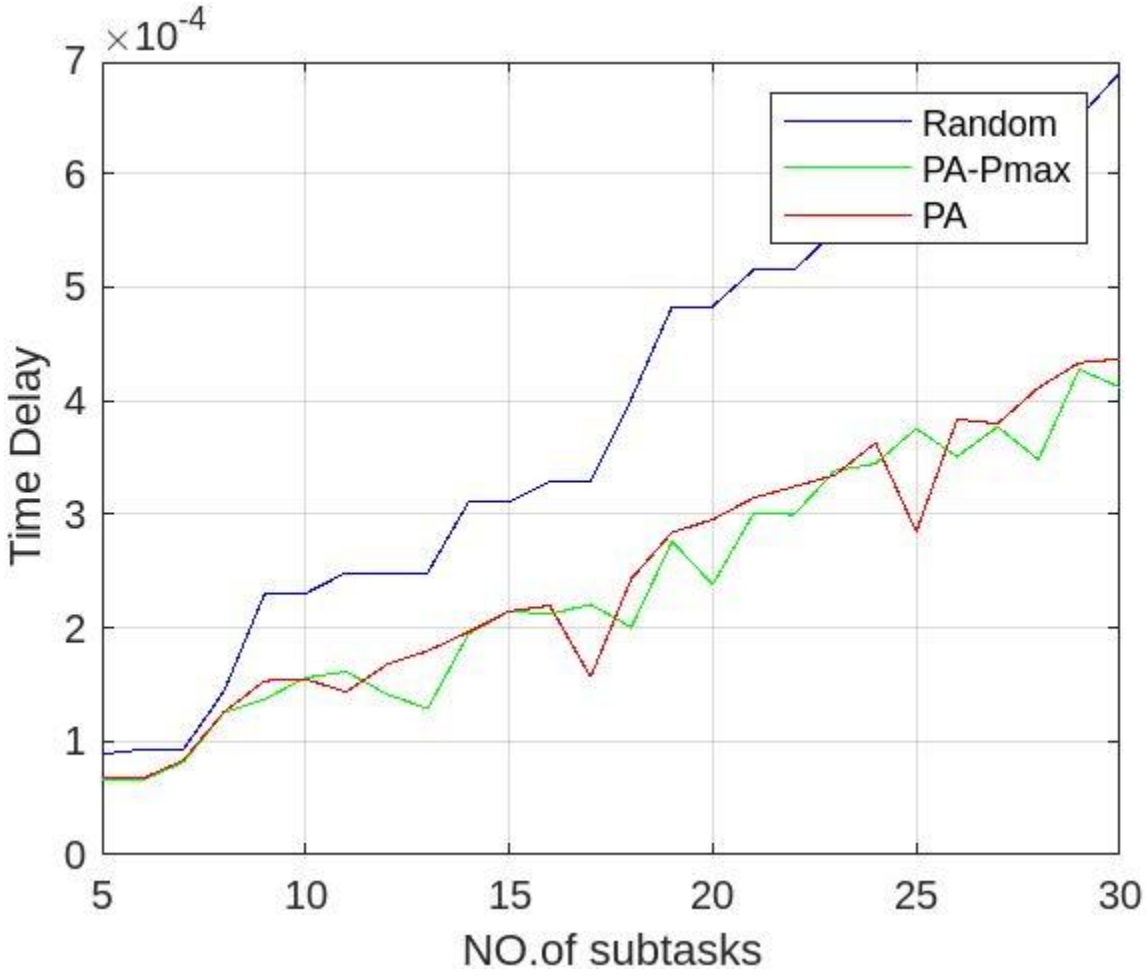
## Work Done

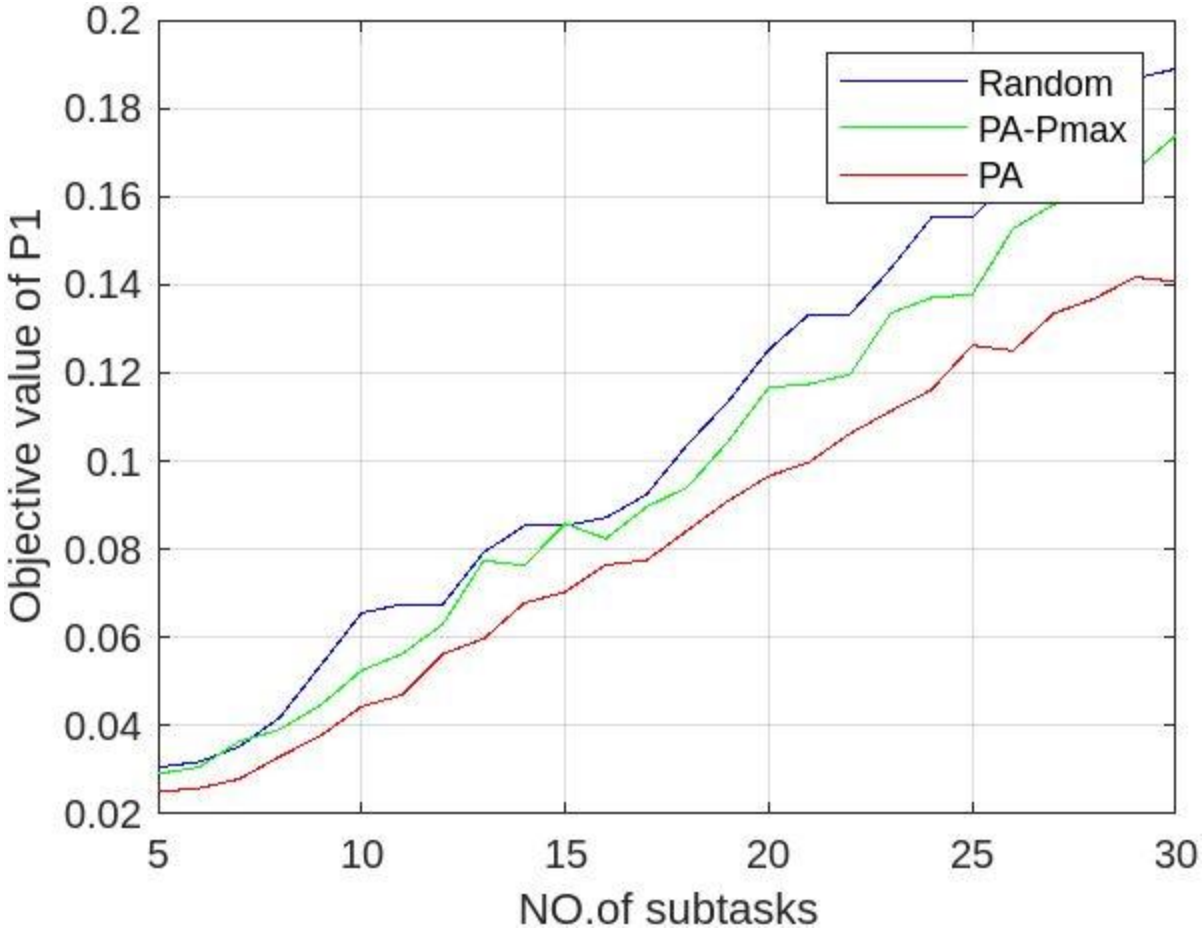
1. The `subtaskAllocation()` function takes in six arguments: `D`, `C`, `VL`, `VK`, `pmax`, and `psi`.
2. The `N` variable is initialized to the length of the `D` array, which represents the number of subtasks.

3. The `F` array is initialized with `N+1` rows and `VL+1` columns. The rows correspond to the subtasks, and the columns correspond to the computation capacities of the mobile device.
4. The `L` and `K` variables are initialized to zero.
5. The first loop initializes the first row of the `F` array to zero.
6. The second loop uses dynamic programming to fill in the rest of the `F` array. For each subtask `i`, the loop checks if the current computation capacity `k` is greater than or equal to the required computation capacity `D[i-1]*C[i-1]`. If so, it calculates the maximum data size that can be computed for the subtask based on the previous row of the `F` array and the current subtask's data size and computation requirement. It updates the current cell of the `F` array with this maximum data size.
7. The `L` and `K` variables are calculated by backtracking through the `F` array. Starting from the bottom-right cell of the `F` array, the algorithm checks whether the maximum data size in the current cell was calculated using the previous row's value. If so, it adds the data size of the current subtask to the `L` variable and subtracts the computation capacity used from the `VL` variable. Otherwise, it moves up to the previous row.
8. The `K` variable is calculated by subtracting the total data size allocated to the mobile device (`L`) from the total data size of all subtasks (`VK`).
9. The `M` variable is calculated by subtracting the total computation capacity used by both the mobile device and the edge server (`K + L`) from the maximum interference threshold `psi`.
10. The `gamma` variable is calculated by dividing the maximum transmit power `pmax` by the sum of the product of the allocated computation capacities and the corresponding interference threshold. It is then multiplied by 100 to convert it to a percentage.
11. The function returns an array of integers containing the values of `M`, `K`, `L`, and `gamma`.
12. The `main()` function sets up sample input values for `D`, `C`, `VL`, `VK`, `pmax`, and `psi`.

13. The `subtaskAllocation()` function is called with the sample input values, and the returned array of integers is stored in the `result` variable.

## Results:-






## Conclusion:-

proposed a subtask pre-allocation algorithm and transmission power allocation scheme for a D2D-assisted MEC system with multiple independent subtasks. The proposed algorithm is designed to minimize the energy consumption and delay in the system, while considering partial task offloading decisions. By leveraging the knapsack problem and alternating optimization with convex optimization, our algorithm achieved significant improvements in reducing the weighted sum of delay and energy consumption, compared to the traditional FOSA algorithm.

Our study provides an effective solution for optimizing resource allocation in D2D-assisted MEC systems. In the future, we plan to investigate the impact of user

mobility on the weighted sum of energy consumption and latency, and extend our work to systems with multiple users.

## References:-

X. Yang, H. Luo, Y. Sun and M. S. Obaidat, "Energy-efficient collaborative offloading for multiplayer games with cache-aided mec", *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1-7, Jun. 2020.

Y. Kai, J. Wang and H. Zhu, "Energy minimization for d2d-assisted mobile edge computing networks", *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1-6, May. 2019.