Mathematical Modelling of Engineering Problems
Vol. 9, No. 3, June, 2022, pp. 668-674
Journal homepage: http://iieta.org/journals/mmep

Intrinsic Profit Maximization of the Offloading Tasks for Mobile Edge Computing with Fixed Memory Capacities and Low Latency Constraints Using Ant Colony Optimization

Harinandan Tunga[1*], Samarjit Kar[2], Debasis Giri[3]

[1] Department of Computer Science & Engineering, RCC Institute of Information Technology, Kolkata 700015, India
[2] Department of Mathematics, National Institute of Technology, Durgapur 713209, India
[3] Department of Information Technology, Maulana Abul Kalam Azad University of Technology, West Bengal 741249, India

Corresponding Author Email: harinandan.tunga@gmail.com

https://doi.org/10.18280/mmep.090313

Received: 5 December 2021
Accepted: 13 June 2022

Keywords:
Ant Colony Optimization, efficiency tasks offloading, mobile edge computing servers, Multiple Knapsack, user equipment

ABSTRACT

Artificial intelligence and the Internet of Things (IoT) have resulted in more computationally demanding and time-sensitive applications. Given the limited processing power of current mobile computers, there is a need for on-demand computing resources with minimal latency. Edge computing has already made a significant contribution to mobile networks, enabling the distribution, scaling, and faster access of computational resources at network margins closer to users, especially in power-constrained mobile devices. Offloading tasks efficiently on the Mobile Edge Computing Server (MECS) is an important part of our proposed method. We propose a method of offloading multiple tasks for Mobile Edge Computing servers that require fixed memory capacities and low latency. We calculate the optimum cumulative intrinsic profit of the number of offloaded tasks efficiently using the Ant Colony Optimization (ACO) model, which is flexible and versatile in the context of real-time applications.

1. INTRODUCTION

The concept of mobile edge computing (MEC) was introduced in the 5th generation of mobile communications technology, where computing resources such as mobile edge computing servers (MECS) are brought near users at the edge of the network. The offloading of tasks from user equipment (UE) to these high-performance servers through access networks reduces latency, ensures greater scalability, minimizes energy consumption, and eases distribution, but also puts more pressure on the MECS. The MEC is designed as a decentralized cloud computing environment and the servers have different capacities and properties. Due to the limited capacities of each server, allocating services and tasks to these distributed servers using an efficient algorithm is an effective solution to optimize the services of MEC and mitigate unnecessary lags, as summarised in Figure 1.

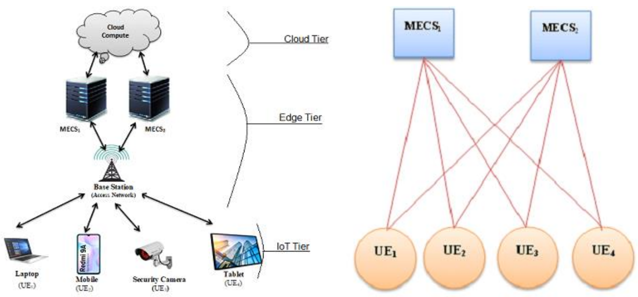


Figure 1. Schematic diagram of Edge Computing Architecture

The efficiency of MECSs and the UEs depends, among other parameters, on the optimal allocation of offloaded tasks to MECS. This paper discusses the problem of the allocation of tasks to MECs. The objective is to maximize the profits intrinsically related to the task by allocating to the MECS or rejecting them altogether based on some constraints discussed later.

ACO algorithms are used to approximate or solve hard combinatorial optimization problems. A multi-agent system is one in which low-level interactions between multiple agents (e.g., artificial ants) produce complex behavior in the entire colony. An ACO algorithm is based on the pheromones that are deposited by ant colonies on the ground (called pheromones). The presence of more pheromone on a particular path increases the chances of the ant choosing it. Our proposed model will address how the ACO algorithm behaves exactly as the artificial ants are represented.

It is shown in section 3 that the resource sharing model of the offloading could be reduced to a simple Multiple Knapsack Problem (MKP). An MKP entails allocating n items, which have some inherent profit and consume some resources, to m Knapsacks. Optimal allocation maximizes profit while not exceeding resource constraints. As MKP is NP-complete, it cannot be used in real-time applications. In this case, Ant Colony Optimization (ACO) is used as a heuristic algorithm.

However, ACO in its original form is not suitable for subset problems such as MKP as its early applications were for ordering problems such as the Travelling Salesperson Problem. So, we have modified the algorithm according to the need of this model.

The main contributions of these works are: namely, the advantages of computational offloading with respect to profit maximization and latency reduction when the ME can offload its computational tasks to MECS; considering the analytical contingency, our proposed algorithms are based on solving Multiple Knapsack problems using Ant colony optimization; according to the results, the ACO-based approaches can achieve close to optimal performance and can reduce latency to a suitable level.

The rest of the paper is organized as follows. Our research is divided into five sections: Literature Review in section 2, Problem Formulation in section 3, Solution Methodology in section 4, which introduces the Ant Colony Optimization System for Multiple Knapsack Problems, followed by numerical results in section 5 and concluding the work in section 6.

2. LITERATURE REVIEW

Leguizamon and Michalewicz [1] proposed a new class of ACO algorithms for the subset problem, which incorporated the computational study of the Multiple Knapsack Problem (MKP) to demonstrate its inherent potential. Model-based searches like ACO are likely to be associated with model biases in decision making. The effectiveness of such biasness was carefully observed and represented by Fidanova [2] based on two non-identical pheromone models. A pheromone model has been used to solve the Multiple Knapsack Problem (MKP), and the results show how useful it is for achieving quality solutions. Based on integer linear optimization, an optimized offloading algorithm was proposed by Khan [3], that allows the selection of execution modes between local execution, offloading execution and dropped tasks for all mobile devices. The NP-hard problem and its allied methods for deployment, resource sharing, load balancing, and fairness practices on multi-user 5G mobile networks are expounded on by Ketyko et al. [4] and NP-hard problems such as multichannel wireless interference can be solved in a centrally optimized way for mobile-edge cloud computing using multiuser offloading. In order to achieve better efficiency, game theory was employed to compute the offloading in the distributed environment [5]. Cena et al. [6] introduced a new version of the ACO model for the Multiple Knapsack Problem (MKP). Guo et al. [7] proposed a flexible framework of offloading methods based on array signal processing for MEC networks. Each antenna was responsible for performing some computations tasks per user. Instead of performing the tasks at the user end with limited computational resources and limited resource capabilities, it could be computed at some competent neighboring computational access points (CAPs), compromising with transmission cost. The costs of the system were calculated using computational price, energy consumption, and latency. The proposed ACO method was able to reduce the system expense by randomly visiting each CAP to obtain the final results. A difficult challenge for executing applications remotely on a mobile device in MCC is computation offloading. Bao et al. [8] addressed this challenge with an ACO based solution with low computational complexity. It can be easily implemented in practice. Zakaryia studied a task scheduling method using mobile edge computing (MEC) with multiple base stations (BS), each consisting of a MEC server, that guides multiple latency-sensitive user equipment (UE) in computing [9, 10]. Sheng et al. [11] also have worked on computing offloading techniques in mobile edge computing. Lee and Bau's work [12] beautifully explains an ant colony optimization method for solving Multidimensional Knapsack problems.

3. PROBLEM FORMULATION

We propose to model the problem of task offloading as a multi-dimensional knapsack (MKP) problem, intending to maximize the cumulative intrinsic profits of the offloaded tasks for servers with fixed memory capacities and minimum delay. This model consists of multiple tasks derived from a single UE and each task can be offloaded to different MECS with different performance specifications. Both UE and MECS are connected via an access network that provides the same bandwidth for all tasks during offloading.

The following symbols with meaning are given in Table 1, which have been used in the problem formulation and solution methodology.

Table 1. List of symbols with description

Symbol	Meaning
N	the set of User Equipment (UEs).
M	the set of Mobile Edge Computing Servers.
b_n	the uploaded tasks and the downloaded results are both measured in bytes.
d_n	the number of CPU cycles.
L	the set of possible computation locations.
o(n): $N \rightarrow L$	the description of a specific offloading setup of UE_N for location L mapping.
$D_{n,l}$	a constant amount of processor time, which is constant throughout the decision-making process.
CPU_n	the number of CPU cycles required by the user equipment n to complete the tasks.
C_m	the capacity of m^{th} Knapsack.
P_i	the profit for i^{th} UE.
$I_{\{\}}$	a function that equals 1 when the statement within the brackets is true, and otherwise 0.
X_{ij}	1, if the item i is in the jth Knapsack, otherwise 0 is returned.
b_i	In the solution of the item i at time t = 0, it represents the number of ants that were present.
N_a	the total number, in a population of ants.
$\tau_i(t+N_{max})$	the Trail/path intensity at time $t+N_{max}$.
P	is the evaporation coefficient.
N_{max}	In some ants, N_{max} is the maximum number of items they can add to a solution.
L^k	the profit obtained by the k^{th} ant.
$allowed_k$	the set of items x is not taken into consideration by the k-th ant, and the solution still satisfies all constraints if some of them are added.
$u_j(k)$	the sum of all the items visited by the k^{th}ant.
δ_{ij}	When the item i is added to the solution, δ_{ij} represents the tightness of item i on constraint j.
$\eta_i(t,k)$	thepseudo-utility for MKP.
α,β	It allows a user to control how much importance is given to trial versus heuristic.
$\Delta\tau_i$	the change in pheromone.
$tabu_k$	the stores the set of items traversed by the k^{th} ant.
NC	the number of iteration counts of the proposed algorithm.
NC_{MAX}	the maximum number of iteration counts permitted for the proposed algorithm.

There are N ={1, 2, ... , n} as a set of user equipments (UEs) and M ={1, 2, . . . , m} as a set of MECSs. Every UE has a user equipment computation task that is either computed by the UE itself or offloaded to the MECS; di is the CPU cycles count for the ith task, qi is the memory required for each task i. Let L = M + {0} represents the set of possible locations for computation. Let l∈L indicates an actual location; l = 0 indicates a UE executing processing, l∈M offloading to one of the MECSs. The function o(n): N → L gives the location L to the UEn mapping, describing a specific offloading setup. While offloading tasks, users encounter latency in real scenarios. The latency of the access network is formulated as follows according to the study [1]:

The offloading setup defines latency (t_n) as the sum of two components: transfer time($t_n^{transfer}$) and execution time($t_n^{execution}$).

$$t_n = t_n^{transfer} + t_n^{execution} \tag{1}$$

$$t_n^{\text{transfer}} = \begin{cases} \dfrac{b_n}{r_{n,o(n)}} \text{ if } l > 0 \\ 0 \text{ if } l = 0 \end{cases} \tag{2}$$

$$t_n^{execution} = \frac{d_n}{f_{n,o(n)}} \tag{3}$$

where, $f_{n,l}$ represents the CPU speed in cycles/second for UEn at location l; $r_{n,m}$ represents the bandwidth between MECSm and UEn; b_n represents the byte size of both the uploaded tasks and the downloaded results and d_n represents the number of CPU cycles.

Definition: The term UE_n refers to a low-latency user in the case of a specific offloading setup, if $t_n < \tau, \tau \in \mathbb{R}^+$. Here, τ refers to the maximum permissible latency of tasks [1].

This concept of latency has been used later in our model to allow only low latency users to offload tasks.

The Resource Sharing Model (RSM) affects the edge to edge computational resource consumption through the definition of $f_{n,l}$. According to the RSM in the cloud, a user can get a predecided quality of memory (Qn,l), which will remain fixed all through the decision-making process. But, MECSs cannot grant users more than their memory capacity (Cm) allows. fn,l is the CPU speed obtained in cycles/second for UEn at location l. An MECS rejects offloading attempts if it exceeds the memory capacity or if it violates the latency constraints:

$$f_{n,l} = \begin{cases} D_{n,l} & l > 0 \text{ and offloading is admitted} \\ 0 & l > 0 \text{ and offloading is rejected} \\ \text{CPU}_n & l = 0 \end{cases} \tag{4}$$

where, CPU_n is the number of CPU cycles required by the user equipment n to complete the task and $D_{n,l}$ is the predetermined amount of processor time constant throughout the decision making process.

Definition: A user is said to be offloaded if o(n) > 0; o(n): N →L describes a specific offloading setup, where UEn is the destination and L is the offload location [1].

In this setup, there are many goals that can be optimized, but we consider that each task has an intrinsic profit Pi that needs to be maximized. Mathematically, enumerated as Z is Maximum offloaded (Max_o).

$$Z = \text{Max}_o \sum_{i=1}^{N} g(i), \text{ where } g(i) \begin{cases} 0 & o(i) \leq 0 \\ P_i & o(i) > 0 \end{cases} \tag{5}$$

The problem can be formulated as follows when taking the capacity of each server into account

$$Z = Max_o \sum_{i=1}^{N} P_i \tag{6}$$

$$Subject\ to \quad o(n) \in L \qquad \forall n \in N \tag{7}$$

$$\sum Q_{i,l} l_{\{o(i)=l\}} \leq c_l \quad \forall l \in M \tag{8}$$

$$t_i \leq \tau \tag{9}$$

where, $l_{\{\}}$ represents the function which is equal to 1 if the statement in the parentheses is true, and otherwise 0. Every server combination can effectively include the UEs that are allocated to that specific server only. In this way, the resulting problem closely resembles the Multiple Knapsack Problem (MKP).

The optimization problem can be expressed as follows in the MKP framework:

Objective Function (Total cumulative intrinsic profits Maximize):

$$Z = Max \sum_{i=1}^{N} \sum_{j=1}^{M} P_i x_{ij} \tag{10}$$

Subject to

$$\sum_{i=1}^{N} Q_i x_{ij} \leq C_j \ , \forall j \in M \tag{11}$$

$$\sum_{j=1}^{M} x_{ij} \leq 1, \forall j \in N \tag{12}$$

$$t_i \leq \tau \tag{13}$$

$$x_{ij} \in \{0.1\} \ \forall i \in N, \forall j \in M \tag{14}$$

The objective function (10) maximizes the total cumulative intrinsic profit of offloaded tasks; Constraint (11) specifies that the cumulative sum of all memory requirements of tasks assigned to a server does not exceed the server's memory capacity; Constraint (12) signifies that each task can be assigned to a single server only; Constraint (13) determines the low latency that each task must be low latency; Constraint (14) specifies that the if x_{ij} is 1 in the Knapsack j, otherwise 0 in item i.

An MKP model involves a set of elements (UEs) with their associated weights (requested memory capacity) and profit, as well as a set of Knapsacks (MECSs) with capacity (maximum memory capacity), and exploring to find the ordering (offloading setup) with the maximum profit.

4. SOLUTION METHODOLOGY OF THE PROBLEM

This section introduces our proposed optimization framework for determining an optimal task allocation decision that minimizes latency and maximizes the intrinsic profits of all tasks.

4.1 Ant Colony Optimization (ACO) model

The ACO model explores the minimum cost of the path on a weighted graph and uses artificial ants to achieve this. An artificial ant behaves similar to a real ant in that it deposits pheromones along its path and chooses its path based on the probabilities of concentration that have been previously laid out. It is possible to create pheromone trails to simulate artificial evaporation to make the model more realistic. We can find optimal solutions for complex problems using a population-based metaheuristic using this proposed model.

4.2 ACO model based edge computing

For the MKP solution, the model of an ant colony has to be adapted because the purpose of the ant here is not only to reduce the cost of the route but also to find the best solution so that the profit of the path is maximized where all resources are satisfied.

Let bi be the number of ants in the solution item 'i' at time t = 0 and suppose Na = $\sum_{=1}^{n} b_i$ is the total number of ants. Since MKP is a subset problem, it is necessary to estimate the path intensity and path visibility so that it can be calculated in a slightly different way for the absence of a clear path. We define τi(t+Nmax) as the path intensity of the item 'i' at the time (t+Nmax), given by:

$$\tau_i(t+N_{max}) = \rho\ \tau_i\ (t) + \Delta\ \tau_i(t,t+N_{max}) \tag{15}$$

where, ρ represents evaporation coefficient and N_{max} represents the maximum number of items some ants add to a solution.

In τi(t, t+ Nmax) =$\sum_{k=1}^{Na}$ Δτik (t,t+ Nmax), where Δτik (t,t+ Nmax) represents the amount of substance per unit length of path (pheromone in real ants) laid on item 'i' between the time 't' and t+Nmax by the kth ant, and is calculated as follows:

$$\Delta\tau_i^k(t, t+N_{max}) = \begin{cases} \frac{L^k}{Q}, \text{ if k}^{\text{th}} \text{ ant includes item 'i'} \\ 0, \text{ otherwise} \end{cases} \tag{16}$$

where Q is a constant and L^k is the profit attained by the k^{th} ant at time t = 0, so the intensity of path $\tau_i\,(0)$ is set at a randomly selected value. In the next $(t+N_{max})$ time, the probability of the k^{th} ant selecting item 'i' to complete the $solution_k$ is:

$$P_i(t,k) = \begin{cases} \dfrac{[\tau_i(t)]^{\alpha}[\eta_i(k,t)]^{\beta}}{\sum_{j\in allowed_k}[\tau_i(t)]^{\alpha}[\eta_i(k,t)]^{\beta}} & i \in allowed_k \\ 0 & i \notin allowed_k \end{cases} \tag{17}$$

A solution$_k$ that has the constraints satisfied by all of the allowed$_k$ items is the solution$_k$ if they are added to the allowed$_k$ set. A local heuristic, or pseudo-utility, the parameter η this case. We have chosen η_i for item 'i' as follows:

$$\eta_i(k,t) = \frac{p_i}{\delta_{ij}(k)};\ \bar{\delta}_{ij}(k) = \frac{\sum_{j=1}^{m}\delta_j(k)}{m} \tag{18}$$

$$\delta_{ij}(k) = \frac{D_i}{(c_j - u_j(k)+1)};\ u_j(k) = \sum_{l\in solution_k} D_l \tag{19}$$

where (c_j - $u_j(k)$) is the remainder of the amount which has reached the boundary of constraint j; $D_i \le (c_j - u_j(k))$ and $\delta ij \epsilon (0,1)$ are the tightness of item 'i' on constraint 'j' when adding item 'i' to the solution. As a result of this is that pseudo-utility η_i (t, k) gets larger and $\delta_{ij}(k)$ (tightness average) gets smaller.

By setting the parameters α and β, the user can control the relative importance of heuristics (pseudo-utilities for MKP) compared with trial. As a result, transition probability is a trade-off between pseudoutilities, where items that benefit the user while using fewer resources are more likely to be chosen with a high probability, and trail intensity, where items that are part of many solutions are highly desirable.

The tabu$_k$ list is a data structure that is associated with every ant, so that it can select items more than once, since the tabu$_k$ list keeps track of the items added by kth ant over time 't'.The list also maintains uj(k) (j= 1, 2,..., m) of the necessary computations so that computation times are reduced.

A set is defined as: allowed$_k$ = [j | j * tabu$_k$] and when item j is added to solution$_k$, it satisfies all constraints. The item h_k is selected from the kth tabu list when all ants have added as many solutions as possible. A kth ant with an initial solution of h_k is released after the kth tabu list is emptied.

4.3 Maximize Cumulative Intrinsic Profit of the Task Offloading Algorithm

In the above discussion, we have explained the ACO-based Maximize Cumulative Intrinsic Profit of the Tasks Offloading method in the Mobile Edge Computing through a flowchart which is shown below in Figure 2.

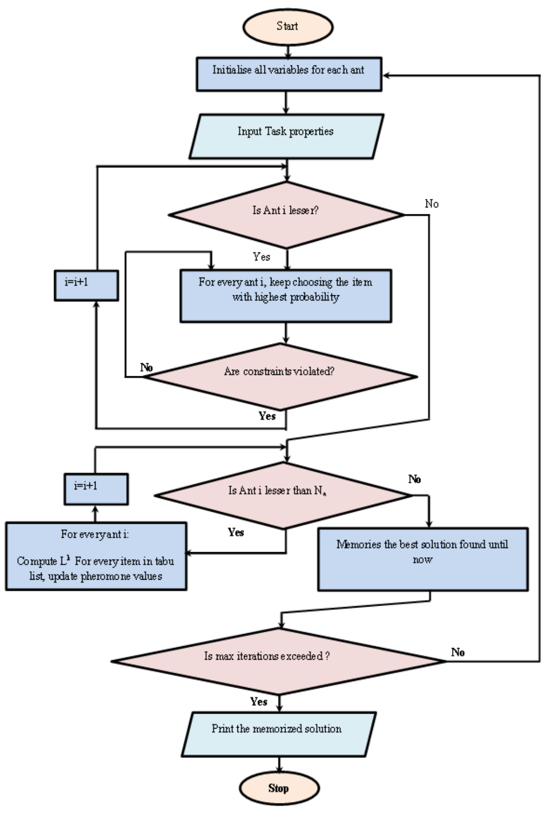


Figure 2. Flowchart of the maximize cumulative intrinsic profit of the tasks offloading

In addition to the above discussion we have developed an ACO-based Maximize Cumulative Intrinsic Profit of the Tasks Offloading algorithm as follows:

Algorithm 1: Maximize Cumulative Intrinsic Profit of the Tasks Offloading

Steps:

1. Set time counter t $\leftarrow$ 0
Set an initial value is τ_i (t) for each item 'i'
Set $\Delta\tau_i(t,t+N_{max})$ $\leftarrow$0 for each item 'i'
b_i is the number of ants choosing item i at time 0
Set tabu list index j$\leftarrow$1
For i$\leftarrow$1 to NDo
For k$\leftarrow$1 to b_i Do
$tabu_k$ (j) $\leftarrow$ i
2. For k$\leftarrow$ 1 to N_a Do
Set tabu list index j $\leftarrow$2
Repeat until all constraints are not satisfied by k^{th} ant
Choose the item 'i' with the highest probability $P_i(t,k)$ given by the equation (17) that doesn't violate
$tabu_k$ (j) $\leftarrow$ i
j $\leftarrow$ j + 1
3. For k $\leftarrow$ 1 to N_a Do
Compute L^k
For s$\leftarrow$ 1 to number of items in $tabu_k$ Do
H$\leftarrow$ $tabu_k$(j)
$\Delta\tau_h$ (t,t+ N_{max})$\leftarrow$ $\Delta\tau_h$ (t,t+ N_{max}) + L^k/Q
j $\leftarrow$ j +1
4. Calculate $\tau_i(t+N_{max})$ according to the equation (15) for each item 'i'
Set time t $\leftarrow$ t + N_{max}
Set $\Delta\tau_i$ (t,t+N_{max}) $\leftarrow$ 0 for each item 'i'
5. Memorize the best solution found up as follows
If ((NC < NC_{MAX}) or (Not all ants find the same solution) Then
$h_k$$\leftarrow$randomly selected item from $tabu_k$
Clear all tabu list
$tabu_k$ (1) $\leftarrow$ h_k
Goto Step 2
Else
Write Best Solution
6. Finish

Following is a step-by-step explanation of the above ACO-based Maximize Cumulative Intrinsic Profit of the Tasks Offloading algorithm for Mobile Edge Computing:

In the **step 1**, time counter 't' is initially set to 0. We assign an initial value of pheromone(τ) on each item and we also assign the change in pheromone ($\Delta\tau_i$) on each item as 0. Then we have assigned the starting item for each ant. The information about the starting item of each ant is stored in the tabu list of the ant.

In the **step 2**, one of the ants among the k^{th} ant is taken into consideration at a time. The probability of visiting all the items in the set of items that the ants have yet to traverse while satisfying all the constraints is calculated for each individual ant. The item with the highest probability and low latency is chosen as the next item to be visited by the ant. The ant's next destination is recorded in its tabu list as the ant moves to its next destination. This process of choosing the next item is continued until there are no items left such that they are not traversed and satisfy all the constraints. This is repeated until all the ants have their set of items they have traversed in order recorded in their tabu lists.

In the **step 3**, one ant out of the k ants is taken into consideration at a time. The total profit, L, obtained from the items chosen by the ant is calculated by adding the profit for each item. For every item in the ant's tabu list, the change in pheromone, $\Delta\tau_i$, is calculated and updated. This process is repeated until the pheromone deposited by all ants on the items in their tabu lists is calculated.

In the **step 4**, the present pheromone value (τ), for each item is calculated by adding the change in pheromone ($\Delta\tau_i$), by the ants obtained from the previous step. Then for each item, the change in pheromone ($\Delta\tau_i$) is again set to 0.

In the **step 5**, if the number of iterations does not exceed the maximum allowable cycle, then the solution (set of items) which was traversed by the maximum number of ants is better than the solution found earlier. Otherwise, it is discarded. If all ants do not reach a consensus regarding a solution, the tabu lists of the ants are emptied and step 2 is followed. If the algorithm cycles are exhausted, the best solution is printed.

5. RESULTS ANALYSIS

A number of parameters have been considered for analyzing the results of the ACO-based Maximized Cumulative Intrinsic Profit of the Tasks Offloading Algorithm in Mobile Edge Computing. Table 2 shows how the test of this algorithm used different input parameters and their values.

Table 2. Different sets of input parameters with their values

S. No.	Input parameters	Input values		
1	Number of iterations (n)	10	20	50
2	Number of ants (N)	100	100	100
3	Evaporation coefficient (ρ)	0.7	0.7	0.7
4	Trade-off between trailing and pseudo-utility factors (α , β)	(0.3, 0.7)	(0.65, 0.35)	(0.9, 0.1)

Table 3. The following data for each test case

Dataset <N,M>	α	β	Ants	Profit(without latency)	Profit(with latency)
<6,2>	0.3	0.7	100	345	265
<6,2>	0.65	0.35	100	345	265
<10,2>	0.3	0.7	100	333	266
<10,2>	0.65	0.35	100	333	266
<10,2>	0.3	0.7	100	452	407
<10,2>	0.65	0.35	100	452	407

The dataset size is <N, M>, where N is the number of tasks and M is the number of servers. With the parameters α and β, we contrast the importance of trial and heuristic, the number of ants, and the maximum profit for the data as provided with the dataset. and our algorithm calculates the maximum profit considering latency.

We consider an interconnected network of UEs and MECSs. Each UE owns a channel with 12 bytes/s transfer speed and $\boldsymbol{\tau}$ which is 2 seconds. All of these have been chosen randomly.

Each test case has been tested for three different sets of pseudo-utility while keeping the number of ants constant. Naturally, profit, when latency limiting is not considered, is greater than when considered. As is clear from the results, changes in the value of parameters don't affect the results in the test cases as shown in Table 3.

To calculate running time and space consumed, a benchmarking test was performed on a system with the following specifications: CPU: AMD Ryzen 3 3250U, 2.6GHz; RAM: 12GB; HDD: 1 TB; Number of ants were incremented in steps and time and space requirements were duly noted for three randomly generated datasets. The number of ants (Na) = 10, comparing the time and space consumption of different data sets is shown graphically in Table 4. The number of ants (Na) = 50, comparing the time and space consumption of different data sets is shown graphically in Table 5. The number of ants (Na) = 100, comparing the time and space consumption of different data sets is shown graphically in Table 6.

Table 4. The Time and Space Consumption of different problems with no. of ants (N_a) = 10

Dataset <N,M>	**Time taken in seconds**	**Space consumed in MB**	**Compare of Time and Space Consumption for 10 Ants of different problem size in Graphically**
<100,50>	14.18	12.67	
<1000,50,>	100.30	14.70	
<1000,100>	886.94	17.30	

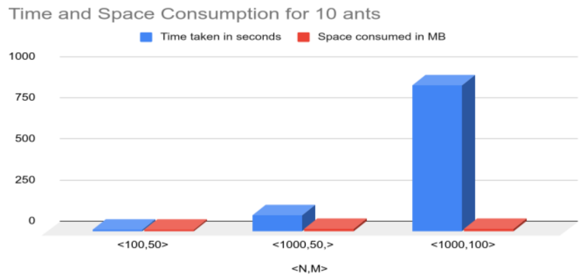


Table 5. The Time and Space Consumption of different problem with no. of ants (N_a) = 50

Dataset <N,M>	**Time taken in seconds**	**Space consumed in MB**	**Compare of Time and Space Consumption for 50 Ants of different problem size in Graphically**
<100,50>	32.54	12.85	
<1000,50>	1795.07	17.10	
<1000,100>	3668.03	19.25	

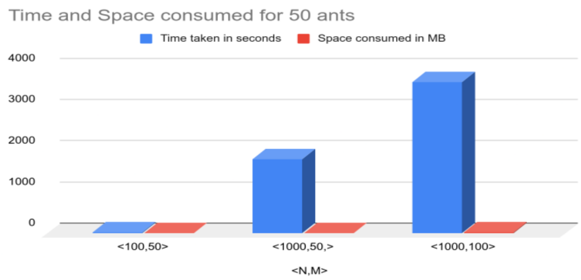
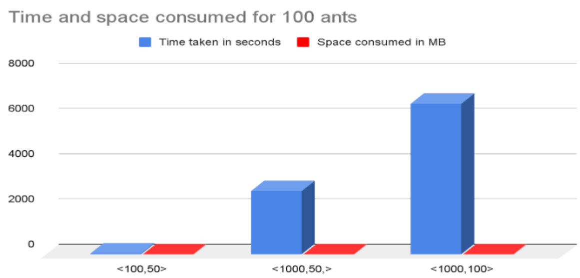


Table 6. The Time and Space Consumption of different problem with number of ants (N_a) = 100

Dataset <N,M>	**Time taken in seconds**	**Space consumed in MB**	**Compare of Time and Space Consumption for 50 Ants of different problem size in Graphically**
<100,50>	49.2	12.85	
<1000,50>	2795.07	17.8	
<1000,100>	6668.03	20.25	



Time and space taken by the algorithm increases according to the polynomial complexity of the algorithm. The runtime of the algorithm may be further reduced if it is supported by multithreading or runs on a CPU with a higher clock cycle or both.

6. CONCLUSIONS

This paper proposes an offloading framework that allows a single UE to offload tasks across multiple Mobile Edge Computing Systems (MECS). We have achieved our goal to maximize profits and limit performance and transmission latency. In order to find efficient solutions to the NP-hard nature of overall optimization problems, we presented the ACO-based Efficiency functions for the Maximized Cumulative Intrinsic Profit of Offloading algorithms.

The ACO algorithm performs reasonably as it has not regurgitated any result that doesn't follow the constraints. Our algorithm has shown high accuracy even during the initial iterations. Hence real-time use of this algorithm for load balancing could also be considered. Also, the algorithm could be made more streamlined or accurate using optimal numbers of ants and values of α and β. There are also plenty of methods to model the ant colony optimization algorithm to fit the MKP. These methods can be tested for a comparative study.

ACKNOWLEDGMENT

The author wishes to express his sincere gratitude to the supervisors and to all anonymous reviewers for their help, knowledge sharing, and suggestions for improving the clarity and proficiency of his research work.

REFERENCES

[1] Leguizamon, G., Michalewicz, Z. (2002). A new version of ant system for subset problems. Proceedings of the 1999 Congress on Evolutionary Computation-CEC 99 (Cat. No. 99TH8406), 2: 1459-1464. https://doi.org/10.1109/CEC.1999.782655

[2] Fidanova, S. (2004). Ant colony optimization for Multiple Knapsack problem and model bias. International Conference on Numerical Analysis and Its Applications NAA, 3401: 280-287. https://doi.org/10.1007/978-3-540-31852-1_33

[3] Khan, P.W., Abbas, K., Shaiba, H., Muthanna, A., Abuarqoub, A., Khayyat, M. (2020). Energy efficient computation offloading mechanism in multi-server mobile edge computing-An integer linear optimization approach. Electronics, 9(6): 1010. https://doi.org/10.3390/electronics9061010

[4] Ketyko, I., Kecskés, L., Nemes, C., Farkas, L., Labs, B., Nokia, R. (2016). Multi-User Computation Offloading as a Knapsack Problem for 5G Mobile Edge Computing. 2016 European Conference on Networks and Communications (EuCNC), 2016, pp. 225-229. https://doi.org/10.1109/EuCNC.2016.7561037

[5] Chen, X., Jiao, L., Li, W.Z., Fu, X.M. (2016). Efficient multi-user computation offloading for mobile edge cloud computing. IEEE/ACM Transactions on Networking, 24(5): 2795-2808. https://doi.org/10.1109/TNET.2015.2487344

[6] Cena, M., Crespo, M.L., Kavka, C., Leguizamón, G. (2000). The ant colony metaphor for multiple knapsack problem. Journal of Computer Science and Technology, 1(2). Retrieved from https://journal.info.unlp.edu.ar/JCST/article/view/1019.

[7] Guo, Y.H., Zhao, Z.C., Zhao, R., Lai, S.W. (2020). Intelligent offloading strategy design for relaying mobile edge computing networks. IEEE Access, 8: 35127-35135. https://doi.org/10.1109/ACCESS.2020.2972106

[8] Bao, W.D, Ji, H.R., Zhu, X.M., Wang, J., Xiao, W.H., Wu, J.H. (2016). ACO-based solution for computation offloading in mobile cloud computing. AIMS, 1: 1-13. https://doi.org/10.3934/bdia.2016.1.1

[9] Wang, Z.L., Li, P.F., Shen, S., Yang, K. (2021). Task offloading scheduling in mobile edge computing networks. Procedia Computer Science, 184: 322-329. https://doi.org/10.1016/j.procs.2021.03.041

[10] Zakaryia, S.A., Ahmed, S.A., Hussein, M.K. (2021). Evolutionary offloading in an edge environment. Egyptian Informatics Journal, 22(3): 257-267. https://doi.org/10.1016/j.eij.2020.09.003

[11] Sheng, J.F., Hu, J., Teng, X.Y., Wang, B., Pan, X.X. (2019). Computation offloading strategy in mobile edge computing. Information 10(6): 191. https://doi.org/10.3390/info10060191

[12] Lee, S., Bau, Y. (2012). An ant colony optimization approach for solving the Multidimensional Knapsack Problem. International Conference on Computer & Information Science (ICCIS), pp. 441-446. https://doi.org/10.1109/ICCISci.2012.6297286