Genetic Algorithm Report for Task Offloading Optimization

# 1. Problem Overview

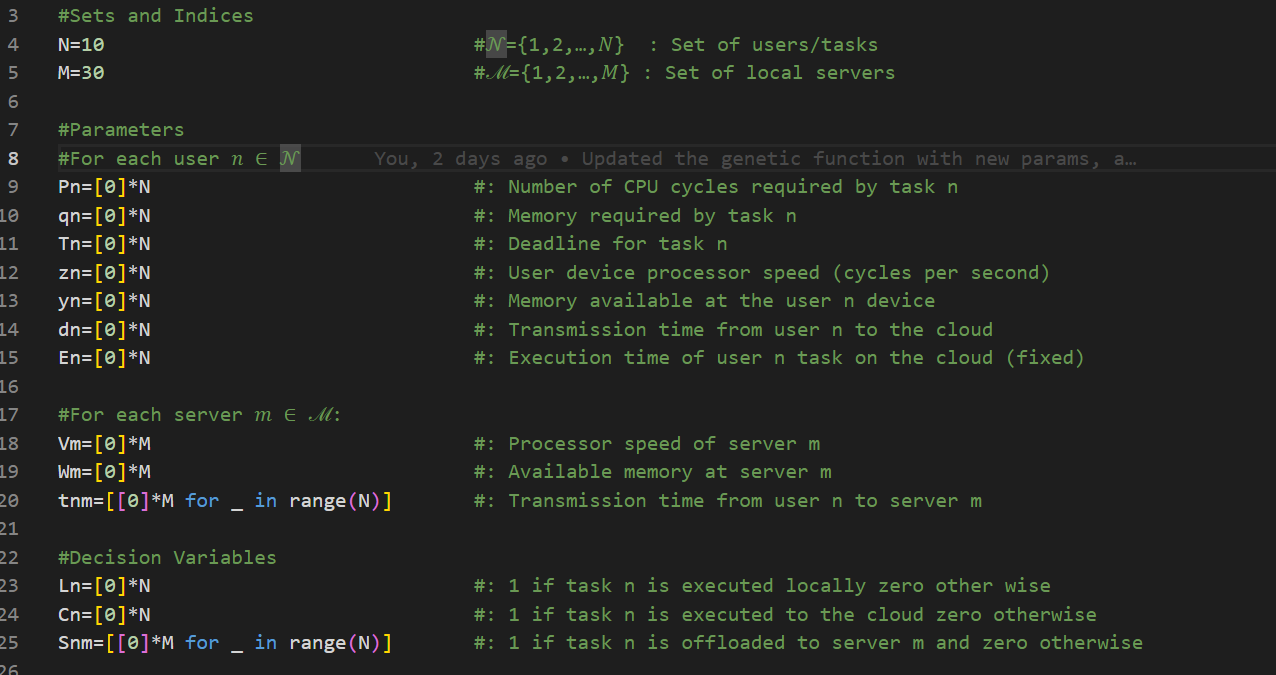
This code optimizes the assignment of tasks from N users to either:

* Local execution on user devices,
* Offloading to one of M local edge servers, or
* Offloading to the cloud.

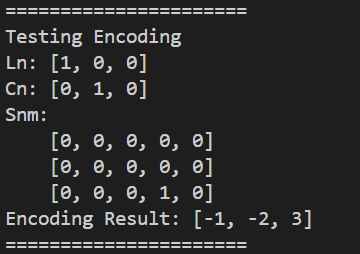
The objective is to **minimize the average task completion time** while satisfying all constraints.

# 2. Genetic Algorithm Components

**Parameters**

****

**Encoding and Chromosome Structure**

* Each **chromosome** is a list of size N (number of tasks).
* A gene value:
  + -1: Task executed **locally**.
  + -2: Task executed in the **cloud**.
  + 0..M-1: Task offloaded to **server m**.

**Fitness Function**

* **average\_task\_completion\_time(...)**: Computes average time per task based on:
  + Local execution time
  + Cloud transmission + execution time
  + Server offloading time (transmission + load-based processing)

# 3. Algorithm Steps

1. Encoding: -1 for local, -2 for cloud, 0 to M-1 task
2. select population length (L)
3. select top K% fittest (elite) based on task completion time
4. These K will be passed to the next generation.
5. From these K selected population, we will generate (100-K) % of the population children
6. From the K values, randomly select 2 random parents.
7. From these parents, perform cross selection based on a random threshold, CHECK VALID? If no repeat.
8. perform mutation, CHECK VALID? If yes, accept it in the next population, if no, repeat.
9. repeat steps 5 to 8, until filling the population.
10. Check stopping criteria, stop if met, otherwise repeat steps 3 to 10.

# 4. Crossover and Mutation Details

**Crossover Type:**

* **Single-point crossover**
* Customizable via: random.randint(1, N-1) (location is **Random**)

**Mutation Type:**

* **Random replacement** mutation
* Randomly reassigns a task to a different valid assignment.
* Adjustable via: mutation probability (prop).

Both operators have **validation checks** to reject infeasible chromosomes.

# 5. Parameter Adjustability

| **Parameter** | **Description** | **Adjustable?** |
| --- | --- | --- |
| population\_size | Number of chromosomes in population | ✅ Yes |
| K\_percent | Top % selected as elite parents | ✅ Yes |
| max\_iterations | Max number of GA iterations | ✅ Yes |
| early\_stop | Stop if no improvement for X iterations | ✅ Yes |
| prop | Mutation probability | ✅ Yes |
| crossover\_point | Point of gene split during crossover | Random |
| mutated[idx] | Mutation point is done by random choice | Random |

# 6. Code Snippet Summaries

**🧬 Encoding/Decoding**

def encode(...): return encoded\_tasks

def decode\_chromosome(...): return Ln, Cn, Snm

Encodes/decodes assignment of tasks for easier genetic manipulation.

**📈 Fitness**

def average\_task\_completion\_time(...): return total\_time / N

Objective function minimizing average task time.

**✅ Constraints**

def verify\_constraints(...): return True/False

Ensures memory limits, deadlines, and assignment rules are met.

**🌱 Population Generation**

def generate\_population(...): return population

Random initialization with constraint checking.

**🧬 Crossover**

def crossover\_chromosomes(...): return [child1, child2]

Random single-point crossover producing valid children.

**🧬 Mutation**

def mutate\_chromosome(...): return mutated

Random single gene mutation with certain adjustable probability and feasibility verification.

# 7. Test run

- Random initialization of parameters for 10 tasks and 30 servers.

- the algorithm is run for 500 iter with population size of 50 and k% of 20%

- keeping in mind the default values:

- the algorithm will stop at early stopping of 10 iter with the same optimal sol.

- probability of mutation occurrence is 5%

A screen shot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**🧠 Interpretation of Results**

**📌 Stopping Condition**

* **Iteration 18** was the last generation.
* The algorithm **stopped early** because there was **no improvement** in the best fitness for **10 consecutive generations** (early\_stop = 10).
* **Best fitness** in the last generation: 0.5039 while the **Best** **Overall** **fitness** was **0**.**4278**

**🧬 Final Best Chromosome**

Encoded Best Chromosome: [3, 5, 2, 21, 13, 26, -1, 22, 0, 13]

Each gene (value) here represents how each task is offloaded:

* -1 → Local execution
* -2 → cloud execution
* >=0 → Task offloaded to a specific edge server (e.g., edge server 3)

This chromosome was decoded into the three main decision variables:

**🔍 Decoded Decision Variables**

**Local Execution (Ln):** [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]

* **Task 6** is executed **locally**.

**Cloud Execution (Cn):** [0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

* **Task 7** is offloaded to the **cloud**.

**Edge Server Assignment (Snm):**

A 10x30 binary matrix (10 tasks × 30 edge servers).

* Rows: tasks
* Columns: edge server indices (0–29)
* A value of 1 at S\_nm means **task n is assigned to edge server m**

Example:

* **Task 0** → Edge server 3
* **Task 1** → Edge server 5
* **Task 2** → Edge server 2
* **Task 3** → Edge server 21
* **Task 4** → Edge server 13
* **Task 5** → Edge server 26
* **Task 8** → Edge server 0
* **Task 9** → Edge server 13