ZENITH

10 DAYS OVER KANPUR

By Pool Peshwas

Algorithm design

Initialisation

Losing drones incurs heavy penalties, so our algorithm always ensures drones recharge before battery-critical missions. All drones first move to the nearest charging station and recharge to full. The drones are assigned to their nearest charging stations

Order Assignment

First, all active and unassigned orders are placed into a priority queue. They are sorted primarily in decreasing order of their point value. If two orders have the same value, the one with the earliest deadline is prioritized to minimize the risk of failure.

The algorithm processes one order at a time, starting with the highest-priority one. For this top order, it identifies the "most feasible" drone, which is the one geographically closest to the order's pickup depot.

The core of this system is its reservation logic. When a drone (e.g., Drone A) is identified as the most feasible, the algorithm calculates the mission's required time window (from a start turn t1 to an end turn t2). It then checks Drone A's schedule to see if it is already reserved for any part of this window.

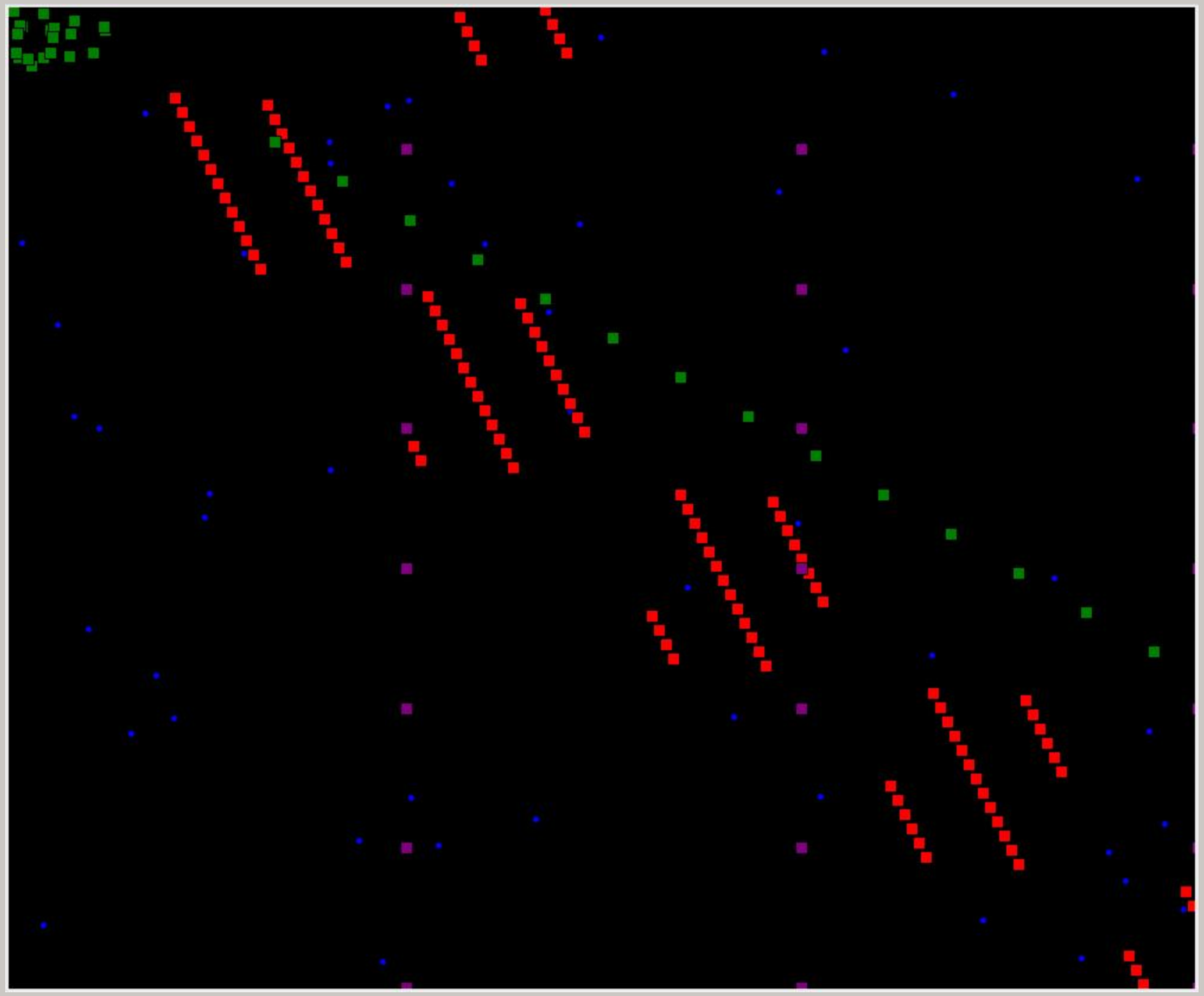
If the most feasible drone (Drone A) is already busy during the required time, the algorithm doesn't discard the order. Instead, it iterates to the next most feasible drone (the second closest) and repeats the availability check. This process continues until an available drone is found for the high-value order. Once a drone is assigned, it is reserved, and the algorithm moves to the next order in the priority list.

Path Planning

For every assigned order, a complete "safety cycle" path is calculated using the Manhattan distance: from the drone's current location to the depot, then to the professor, and finally to the nearest charging station. Before starting this route, a critical battery check is performed. The algorithm ensures the drone's current battery is greater than the total energy cost for the entire trip, including all movements and actions.

Example

If a drone does not have sufficient battery for a full delivery cycle, it is first sent to recharge before starting the mission. For instance, if a drone has 15 battery units but the required energy for a mission is 16, it will go to charge first. This proactive charging strategy is the key mechanism that ensures reliability and prevents any drone from being lost due to battery depletion.



Blue are drones
Red are professors
Green are charging stations
Purple are Depots

Getting the passwords

```python
from Crypto.Util.number import *
from functools import reduce
import random

def rI(b, mod):
    return getRandomNBitInteger(b) + mod

def cP(a, b, m):
    return a ** m + b

def chP(c):
    return isPrime(c)

def gP(m):
    while True:
        a = rI(2048 // m, 1)
        r = rI(m ** 2, random.randint(1, 3))
        p = cP(a, r, m)
        if chP(p):
            return (p, r)
```

```python
def enc(m, e, n):
    return pow(m, e, n)

def cnv(msg):
    return bytes_to_long(msg)

def sm(mb):
    e = 65537
    p, a = gP(mb)
    q, b = gP(mb)
    n = p * q
    return e, n, p, q, a, b

def encrypt():
    e, n, p, q, a, b = sm(4)
    with open("password.txt", "rb") as f:
        password = f.read().strip()
    mLng = cnv(password)
    c = enc(mLng, e, n)
    with open("out.txt", "w") as out:
        out.write(f"n = {n}\n")
        out.write(f"e = {e}\n")
        out.write(f"c = {c}\n")
        out.write(f"secret1 = {a}\n")
        out.write(f"secret2 = {b}\n")

encrypt()
```

Variation of RSA encryption

p = a**4 + r_1

q = b**4 + r_2

a and b are 512-bit

r_1 and r_2 are 16 bit

N = p * q (approx)

N = (a**4 + r_1) * (b**4 + r_2)

S = N ** 1/4 ⟶ S = a * b

r_2 * X^2 - (N - S^4 - r_1 * r_2) * X + r_1 * S^4 = 0

solved for X, find fourth root as X = a**4

compute p = a**4 + r_1 then q