Delivery Scheduling

Consider a delivery scenario where packages of three types - fragile, normal, and urgent - must be transported from a starting point at coordinates (0, 0) to various delivery locations. Each package type incurs different costs and penalties during transportation.

Objective:
Design an algorithm to optimize the delivery order of packages, considering the following criteria:

(1) Minimize Fragile Damage:
Fragile packages have a chance of damage (**X%**) for every kilometer traveled (**Y**), incurring a cost of **Z** for each damaged package. The probability of a package breaking is calculated as follows: $P_{damage} = 1 - (1 - X)^{Y}$
You should calculate P_{damage} for all fragile objects when they arrive at the destination. Then calculate whether the object is damaged or not.

```
distance_covered = sum(distances)
chance_of_damage = package.breaking_chance
p_damage = 1 - ((1 - chance_of_damage) ** distance_covered)
if random.uniform(0, 1) < p_damage:
    print('Package broken')
```

(2) Minimize Travel Costs:
Each kilometer traveled incurs a fixed cost **C**.

(3) Adhere to Urgent Delivery Constraints:
Urgent packages incur a penalty for delivery outside the expected time, penalized by a fixed amount for each minute of delay. The penalty per minute is equal to the fixed costs **C.**

Constraints:

1. You only have one vehicle available.
2. The delivery locations are specified by their coordinates.
3. Routes between all delivery coordinates are available.
4. The driver drives at 60km per hour and takes 0 seconds to deliver the goods.
5. The cost per km is C=0.3.

Package Types:

1. Fragile packages: Have a chance of damage during transportation.
2. Normal packages: No risk of damage during transportation.
3. Urgent packages: Incur a penalty for delivery outside the expected time.

Objective Function:
Minimize the total cost, considering fragile damage, travel costs, and urgent delivery penalties.

Input:
Package information, including type (fragile, normal, urgent) and coordinates of delivery locations.
Use the following script to generate package data (change it to suit your needs - the harder the problem you solve, the better):

```
import random
import pandas as pd

class Package:
    def __init__(self, package_type, coordinates):
        self.package_type = package_type
        self.coordinates_x = coordinates[0]
        self.coordinates_y = coordinates[1]

        if package_type == 'fragile':
            self.breaking_chance = random.uniform(0.0001, 0.01)  # 0.01-1%
chance of breaking per km
              self.breaking_cost = random.uniform(3, 10)  # Extra cost in
case of breaking

        elif package_type == 'urgent':
            self.delivery_time = random.uniform(100, 240)  # Delivery time
in minutes (100 minutes to 4 hours)

def generate_package_stream(num_packages, map_size):
    package_types = ['fragile', 'normal', 'urgent']
              package_stream    =    [Package(random.choice(package_types),
(random.uniform(0,  map_size),  random.uniform(0,  map_size)))  for  _  in
range(num_packages)]
    return package_stream

# Example: Generate a stream of 15 packages in a map of size 60x60
```

```
num_packages = 15
map_size = 60
package_stream = generate_package_stream(num_packages, map_size)

df = pd.DataFrame([(i, package.package_type, package.coordinates_x,
package.coordinates_y, package.breaking_chance if package.package_type ==
'fragile' else None, package.breaking_cost if package.package_type ==
'fragile' else None, package.delivery_time if package.package_type ==
'urgent' else None) for i, package in enumerate(package_stream, start=1)],
columns=["Package", "Type", "CoordinatesX", "CoordinatesY", "Breaking
Chance", "Breaking Cost", "Delivery Time"])
```

Output:
Optimized delivery order that minimizes the total cost.

Evaluation criteria:
Total cost: your algorithm should provide the deliveries at the lowest cost possible.
Reputation: your algorithm should ensure deliveries on time and minimize package breaks to keep the reputation of the delivery company intact.