ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

CS452 - Introduction to the Science and Technology of Services

Fall Semester 2023-2024

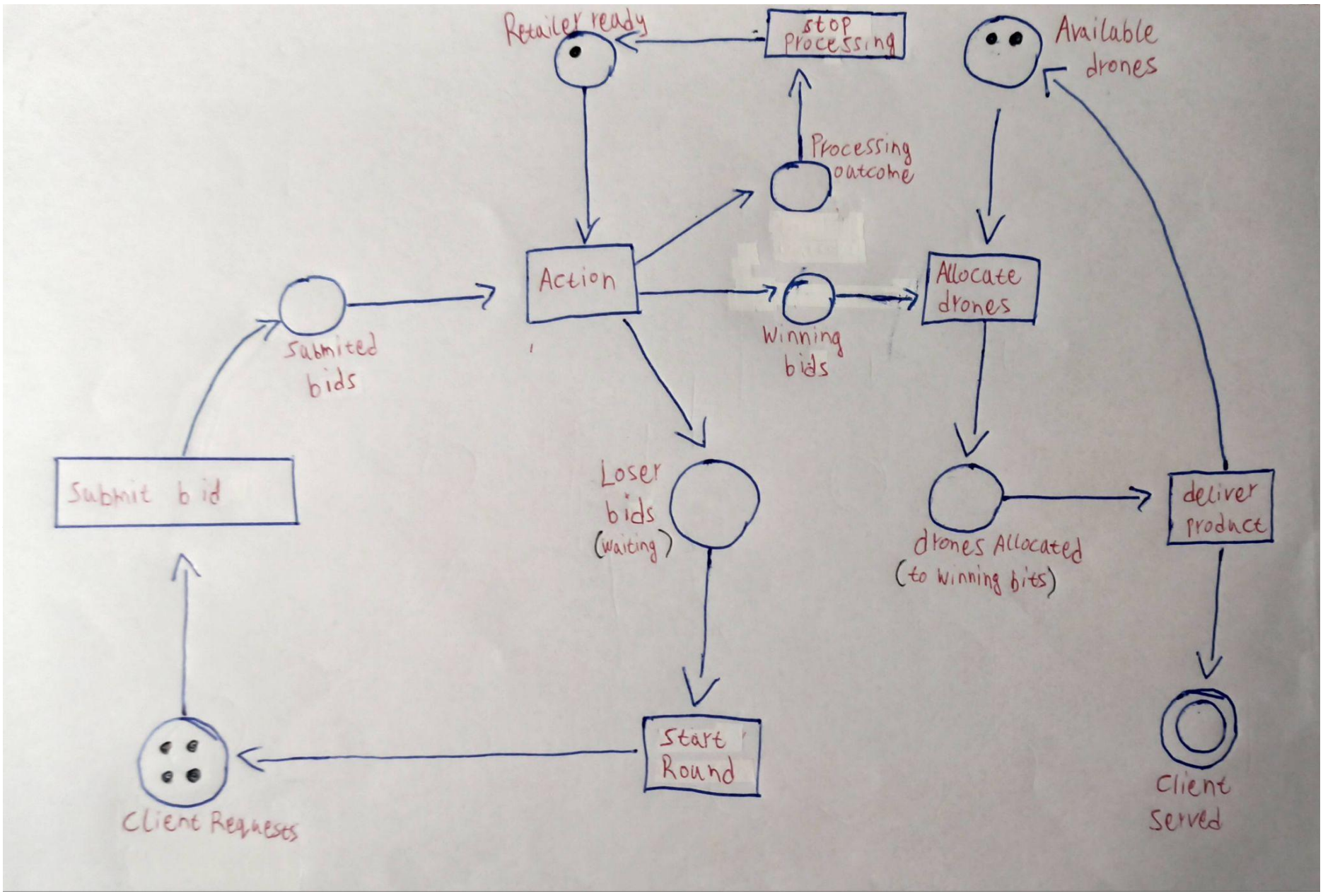
3rd Assignment Report

Κουμάκης Εμμανουήλ
AM : 4281

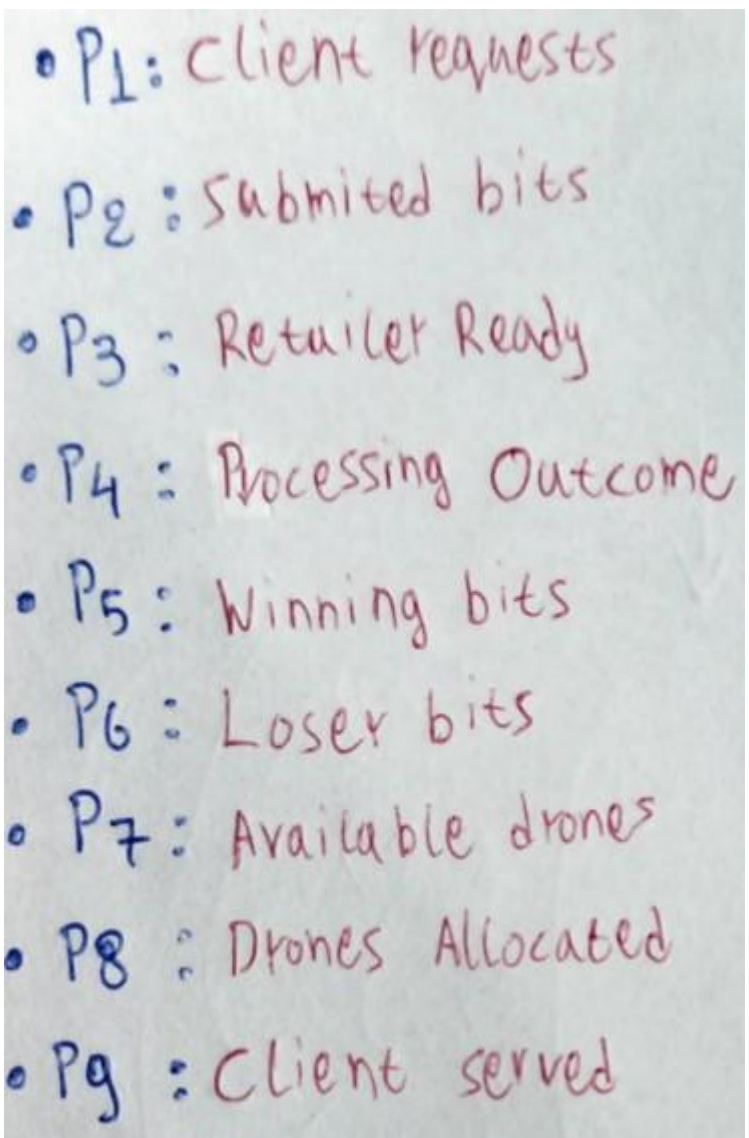
Phase A : Petri net

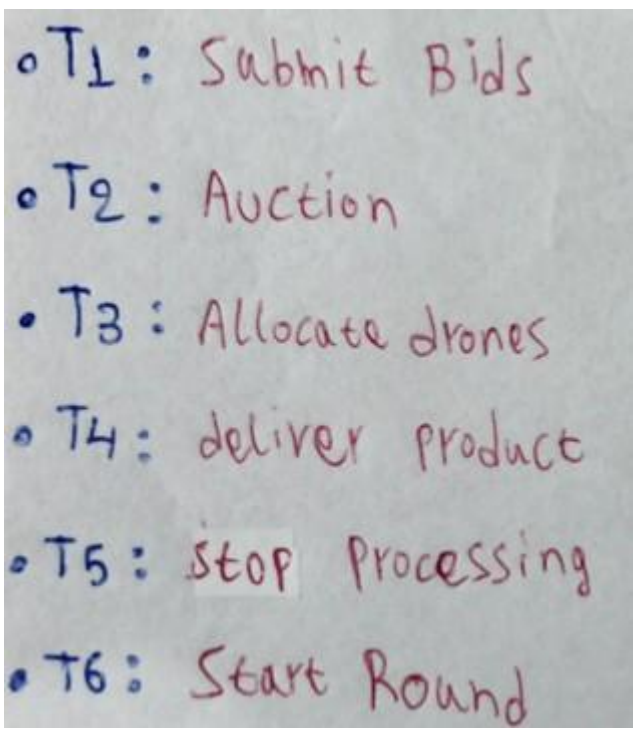
Στην εκφώνηση αναφέρεται ότι “ο χρόνος χωρίζεται σε γύρους και σε κάθε γύρο ο retailer δέχεται ένα fixed αριθμό από requests”. Έτσι για την σχεδίαση του petri net υπέθεσα ότι το action ξεκινάει αφού έχουν κάνει όλοι οι clients submit (clients = requests) τα bit τους, όχι όμως ότι κάνουν όλοι submit ταυτόχρονα.
Επίσης υποθέτω (όπως ειπώθηκε στο φροντιστήριο που πραγματοποιήθηκε) ότι ο δεύτερος γύρος ξεκινάει αφού παραδοθούν τα προϊόντα του πρώτου γύρου (και επιστρέψουν τα drones) καθώς και όταν ο retailer έχει επεξεργαστεί τα αποτελέσματα του πρώτου και είναι έτοιμος να λάβει τα επόμενα requests.

Το petri net του συστήματος :

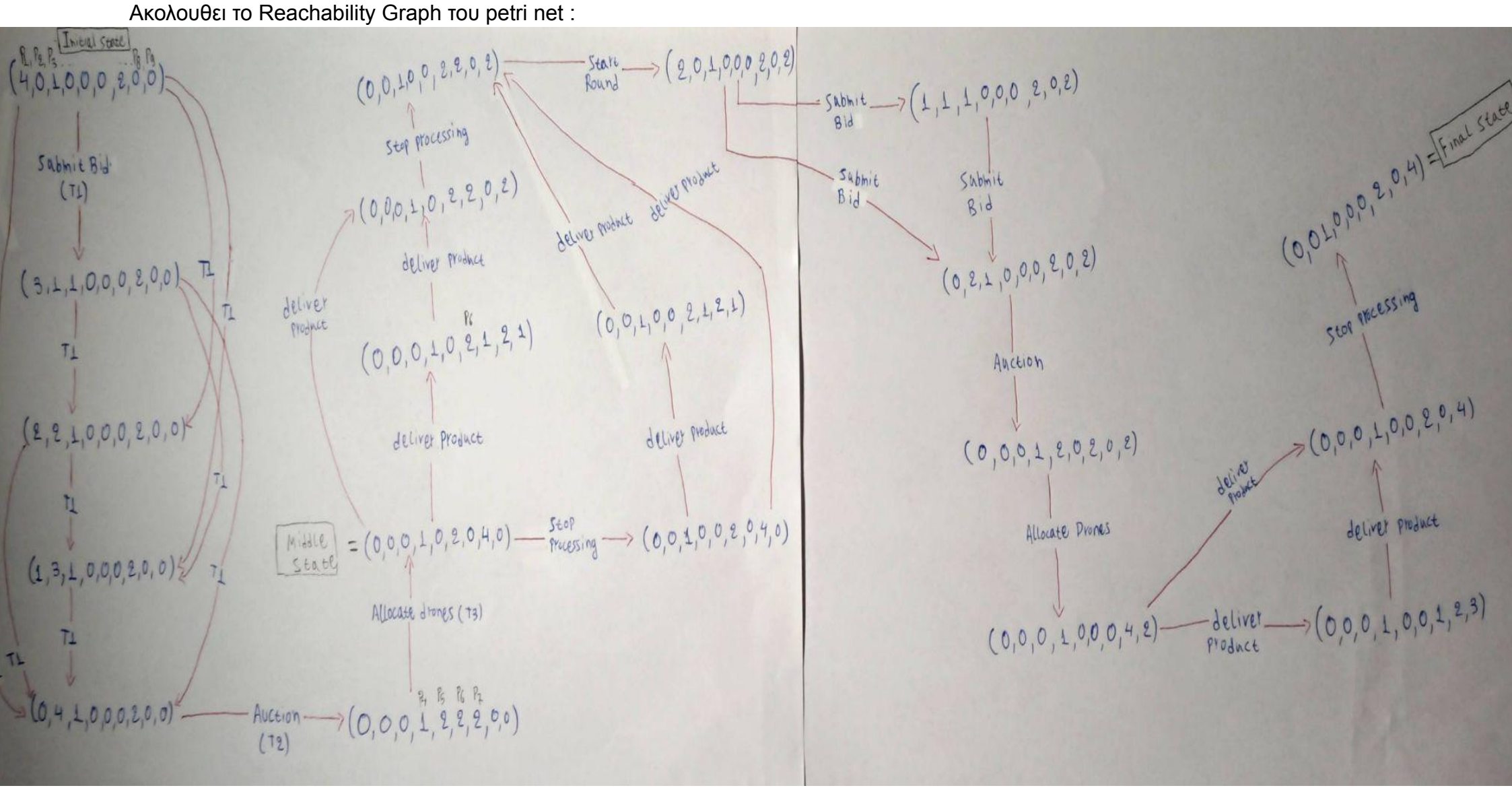


Τα Places & transitions του συστήματος (σκονάκι για το reachability graph) :





Ακολουθει το Reachability Graph του petri net :



(Ο λόγος που υπάρχουν πολλαπλά “submit bit” transitions & “deliver product” transitions αντίστοιχα, είναι επειδή σε ένα ‘ρεαλιστικό’ σενάριο δεν υποβάλλουν όλοι οι clients ταυτόχρονα τα bids τους, καθώς και τα προϊόντα δεν παραδίδονται ταυτόχρονα επίσης. Έτσι το σύστημα μας θα μπορεί να βρεθεί σε όλες αυτές τις διαφορετικές καταστάσεις.)

- Όπως φαίνεται στο Middle state : ο retailer βρίσκεται σε κατάσταση που επεξεργάζεται τα αποτελέσματα του πρώτου γύρου (P4 = processing outcome), Τα 2 loser bids περιμένουν να ξεκινήσει ο επόμενος γύρος (P6), και τα 2 διαθέσιμα drones έχουν ανατεθεί στα 2 winning bids (P8 = allocate drones).
- Στο final state : ο retailer έχει σταματήσει να επεξεργάζεται τα αποτελέσματα του 2ου γύρου και έχει επιστρέψει στο P3, τα drones έχουν παραδώσει τα προϊόντα και έχουν επιστρέψει στο P7 και όλοι οι clients έχουν εξυπηρετηθεί και έχουν φτάσει στο τελικό state P9 = client served.

Phase B

Έχουμε ουσιαστικά να λύσουμε ένα πρόβλημα βελτιστοποίησης για κάθε ένα από τους, retailer, drone provider και clients.
1.Για τον retailer πρέπει να βρούμε ένα είδος δημοπρασίας που να επιφέρει το μεγαλύτερο συνολικό κέρδος (έσοδα από τα bids).
2.Για τους clients πρέπει να βρούμε μια στρατηγική που να προσφέρει το μεγαλύτερο pay-off (το μικρότερο δυνατό ποσό που να μπορεί πληρώσει (bid) για να εξυπηρετηθεί).
3. Για τον drone provider πρέπει να βρούμε μια στρατηγική (σειρά εξυπηρέτησης πελατών) με την οποία θα ελαχιστοποιηθεί ο συνολικός χρόνος εξυπηρέτησης.

Σημείωση : Η ύπαρξη ίδιων ή διαφορετικών ειδών drones δεν αφορά και δεν αλλάζει κάτι στην στρατηγική του retailer ή τον clients. Το μόνο που επηρεάζεται είναι ο scheduling αλγόριθμος που θα χρησιμοποιήσει κάθε φορά ο drone provider.

1) Retailer :

Για να αποφασίσουμε ποιος είναι ο πιο συμφέρων τύπος δημοπρασιας για τον retailer πρέπει να λάβουμε υπόψη τα παρακάτω :

1. Ο χρόνος είναι χωρισμένος σε γύρους και ο κάθε γύρος τελειώνει αφού έχουν υποβάλει όλοι οι clients, που έχουν στείλει request, τα bids τους.
2. Πρόκειται για single unit action (ένα drone θα ανατεθεί σε κάθε νικητή)
3. Λόγο του 1 βολεύει ο κάθε client να επιτρέπεται να υποβάλει μια φορά bid σε κάθε γύρο.

Έτσι κυρίως επειδή ο κάθε client θα μπορεί να υποβάλει το bid του μονάχα μια φορά, καταλήγουμε στο συμπέρασμα ότι πρέπει να χρησιμοποιήσουμε κάποιου είδους sealed bid auction.

Έτσι οι δυνατές επιλογές μας 1. first price sealed bid ή 2. Vickrey Auction (second price).

Όσον αφορά τον καθορισμό τον νικητών, εφόσον έχουμε δημοπρασία, πρέπει να επιλέγονται κάθε φορά οι clients με τα μεγαλύτερα bids (π.χ αν μπορούμε να έχουμε 5 νικητές τότε θα επιλέγονται αυτοί με τα 5 μεγαλύτερα bids).
Πιστεύω ότι το πιο λειτουργικό για το συγκεκριμένο project είναι ο κάθε νικητής να πληρώνει το ποσό το οποίο είχε υποβάλει (το bid του), καθώς το να πλήρωνε ένα ποσό που θα κάνει scale ανάλογα με την τοποθεσία του ή/και το βάρος του πακέτου του, χαλάει το νόημα της δημοπρασίας.

Λαμβάνοντας υπόψη το παραπάνω καταλήγουμε στην επιλογή του first price sealed bid ως το κατάλληλο είδος δημοπρασίας ανάμεσα στα δύο, καθώς μας βολεύει ο κάθε client να πληρώνει και το ποσό που υπέβαλε αρχικά και όχι το ποσό του αμέσως προηγούμενου του (όπως θα ήταν στην δημοπρασία Vickrey).

2) Drone provider :

A) Identical Drones :

Σε αυτό το πρώτο σενάριο όλα τα drones είναι ίδια (όσον αφορά την ταχύτητα και το μέγιστο βάρος που μπορούν να κουβαλήσουν), επομένως οι μοναδικοί παράγοντες είναι η απόσταση από την τοποθεσία του κάθε client και το ποσό που έχει πληρώσει ο κάθε client για την μεταφορά.

Θα ήταν δίκαιο για τον client που έχει πληρώσει το μεγαλύτερο ποσό (bid), να έχει και την μεγαλύτερη προτεραιότητα στην ουρά παράδοσης.
Από την άλλη για τον drone provider το πιο αποδοτικό θα ήταν να εξυπηρετούνται οι clients με βάση την απόσταση τους από την αποθήκη του retailer (πρώτα αυτοί που είναι πιο κοντά).

Για αυτό το λόγο μπορούμε να εφαρμόσουμε μια φόρμουλα που θα βοηθήσει στον καθορισμό μιας αποδοτικής και κατά κάποιο τρόπο δίκαιης σειράς παράδοσης των προϊόντων :

Σειρά παράδοσης = B1 × 10 / Απόσταση(σε km) + B2 × BID

Και επειδή τον drone provider τον ενδιαφέρει περισσότερο να ελαχιστοποιήσει τον συνολικό χρόνο παράδοσης, μπορούμε να δώσουμε μεγαλύτερη βαρύτητα στον συντελεστη B1.

π.χ η φόρμουλα μπορεί να γίνει : Σειρά παράδοσης = 0.75 × 10 / Απόσταση(σε km) + 0.25 × BID

Η φόρμουλα θα υπολογίζεται ξεχωριστά για κάθε client που ‘νικησε’ αυτό το γύρο και θα εξυπηρετούνται πρώτα αυτοί με το μεγαλύτερο συνολικό άθροισμα.

B) Different type drones :

Σε αυτό το δεύτερο σενάριο, το profile του client παραμένει το ίδιο (έχει ως παράγοντες την απόσταση και το bid που έχει πληρώσει για να ‘νικήσει’).
Από την μεριά του drone provider όμως τώρα, πρέπει να υπολογιστεί και ένας τρίτος παράγοντας, το βάρος του κάθε προϊόντος.

Αυτή τη φορά τα προϊόντα που ξεπερνούν ένα βάρος X θα αναθέτονται στα Type A drones (low speed, heavy packages) και τα υπόλοιπα στα Type B.
Τώρα όσον αφορά την σειρά προτεραιότητας εξυπηρέτησης των clients μπορεί να χρησιμοποιηθεί ξανά η φόρμουλα που υπολογίσαμε προηγουμένως, για κάθε μια από τις κατηγορίες των drones.

3) Clients :

Σε ένα ρεαλιστικό σενάριο, ο κάθε client ενδεχομένως να λαμβάνει διαφορετικές αποφάσεις ανάλογα με το profile του. Σύμφωνα δηλαδή με το πόσο γρήγορα θέλουν να παραλάβουν το προϊόν, πόσο οικονομικά ευκατάστατοι είναι για να μπορούν να υποβάλουν υψηλά bids, πόσο μακριά βρίσκονται από την τοποθεσία του retailer (θεωρώντας ότι γνωρίζουν τον τρόπο με τον οποίο λειτουργεί ο provider που περιγράψαμε παραπάνω), πόσο ανέχονται ή αποφεύγουν τον κίνδυνο (risk averse ή risk neutral), πόσα γνωρίζουν για την συμπεριφορά των υπόλοιπων clients και τις στρατηγικές τους, σε ποιο γύρο βρίσκονται (αν έχουν χάσει σε προηγούμενους γύρους ίσως έχουν βγάλει συμπεράσματα, αφού βγουν τα αποτελέσματα, που ίσως τους κάνουν να αλλάξουν συμπεριφορά), και πολλά άλλα…

Γνωρίζουμε ότι το είδος της δημοπρασίας είναι first price sealed bid. Σε ένα τέτοιο είδος δημοπρασίας η κυρίαρχη στρατηγική για τους clients θα ήταν να μειώσουν κάπως τα bid τους (shade bids), από την πραγματική τους αποτίμηση, έτσι ώστε να αποφύγουν το winner’s curse. (στην συγκεκριμένη περίπτωση winner’s curse = ο client πληρώνει περισσότερο από όσο χρειαζόταν για να μπει στην λίστα με τους νικητές που θα εξυπηρετηθούν).
Όμως στο δικό μας σενάριο έχει σημαντική επιρροή στην στρατηγική ενός client ο χρόνος στον οποίο επιθυμεί να εξυπηρετηθεί (πόσο γρήγορα), καθώς αυτοί με τα πιο υψηλά bids θα έχουν κάποιο πλεονέκτημα στην σειρά παράδοσης.

Επομένως ο κάθε client μπορεί να διαμορφώσει την στρατηγική του σύμφωνα με το πόσο πολύ τον ενδιαφέρει η ταχύτητα παράδοσης.

1. Οι clients που θέλουν να παραλάβουν άμεσα το προϊόν τους μπορούν να πληρώσουν (bid) την πραγματική τους αποτίμηση με σκοπό να λάβουν προτεραιότητα στην σειρά παράδοσης.
2. Από την άλλη, οι clients που τους ενδιαφέρει απλά να μπουν στην λίστα με τους νικητές και δεν τους νοιάζει τόσο το να εξυπηρετηθούν πρώτοι, μπορούν να εφαρμόσουν την στρατηγική του bid shading που αναφέραμε ώς κυρίαρχη προηγουμένως.

Phase C

C1 (Retailer's side) :

- First lambda function : Συγκεντρώνουμε και προωθούμε τα bids, #drones κλπ όπως φαίνεται στο screenshot

```python
def lambda_handler(event, context):
    data = {
    "drones": "",
    "clients": []
    }
    # Get number of available drones
    drones = 0
    paginator = DB.get_paginator('scan')
    for page in paginator.paginate(TableName='Drones'):
        for item in page.get('Items', []):
            if item.get('isAvailable'):
                drones += 1
    logger.info(f"Number of available drones: {drones}")
    data["drones"] = drones

    # Get all requests from SQS
    requests = 0
    while True:
        sqs_response = sqs.receive_message(
            QueueUrl=sqs_queue_url,
            AttributeNames=['All'],
            MaxNumberOfMessages=10,
            WaitTimeSeconds=5
        )

        logger.info(f"Received {len(sqs_response.get('Messages', []))} messages from SQS")
        #Stop when queue is empty
        if(len(sqs_response.get('Messages', [])) == 0):
            break


        for message in sqs_response.get('Messages', []):
            logger.info(f"Received Message: {message['Body']}")
            requests+= 1
            data["clients"].append(json.loads(message['Body']))
            logger.info(data["clients"])

            # Delete the message from the queue
            sqs.delete_message(
                QueueUrl=sqs_queue_url,
                ReceiptHandle=message['ReceiptHandle']
            )

    #forward it to second lambda
    resp = lambda2.invoke(
            FunctionName = 'find_optimal_requests',
            InvocationType = 'RequestResponse',
            Payload = json.dumps(data,indent=2)
    )

    return {
        'statusCode': 200,
        'body': json.dumps(f'{requests} requests have been forwarded to lambda2')
    }
```

Διαβάζουμε και αποθηκεύουμε τον αριθμό των διαθέσιμων drones από τη βάση

Παίρνουμε όλα τα μηνύματα, με τα στοιχεία των συμμοτεχόντων, από την ουρά

Προωθούμε τα στοιχεία που συγκεντρώσαμε στην επόμενη συνάρτηση lambda.

- Second lambda function : Βρίσκουμε τα optimal bids (winning clients) όπως φαίνεται στο screenshot

```python
def lambda_handler(event, context):

    logger.info(event)

    winners = {"clients" : []} #optimal bids - winning clients
    all_clients  = {"winners" : [], "losers": []}

    if(len(event['clients']) > event['drones']):
        sorted_clients = sorted(event['clients'], key = lambda x: x['bid'], reverse = True)
        winners['clients'] = sorted_clients[:5]
        all_clients['winners']  = sorted_clients[:5]
        all_clients['losers']  = sorted_clients[5:]
    else: #when #requests are less than the #drones (X)
        winners['clients'] = event['clients']
        all_clients['winners']  = event['clients']

    logger.info(f"Winners : {winners}")
    logger.info(f"All clients  : {all_clients}")

    #forward winning clients (optimal bids) to drone provider
    resp = lambda2.invoke(
        FunctionName = 'send_for_delivery',
        InvocationType = 'RequestResponse',
        Payload = json.dumps(winners,indent=2)
    )

    logger.info('-------Winners forwarded-------')
    logger.info(resp['Payload'].read().decode('utf-8'))

    #forward all requests to other lambda that will notify clients for the results
    resp = lambda2.invoke(
        FunctionName = 'respond_to_clients',
        InvocationType = 'RequestResponse',
        Payload = json.dumps(all_clients,indent=2)
    )
    logger.info('------all clients forwarded--------')
    logger.info(resp['Payload'].read().decode('utf-8'))
```

Βρίσκουμε τα optimal (winning) Bids και διαχωρίζουμε τους winners από τους losers

Προωθούμε τους winners στην επόμενη συνάρτηση lambda που θα ειδοποιήσει τον drone provider

Προωθούμε όλους τους συμμετέχοντες (χωρισμένους σε winners _losers) στην επόμενη συνάρτηση lambda που θα ειδοποιήσει όλους τους clients με τα αποτελέσματα

- Third lambda function : Προωθούμε τα X optimal bids στον drone provider

```python
import json
import logging
import requests

# Enable logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)

provider_endpoint = 'http://ec2-54-197-214-200.compute-1.amazonaws.com:5000/'
headers = {"Content-Type": "application/json"}

def lambda_handler(event, context):

    logger.info(event)

    #just send the winners (a json with their bids, distance etc) to drone provider ec2
    try:
        # Send the POST request
        response = requests.post(provider_endpoint, data = json.dumps(event), headers = headers)

        logger.info(f"Status Code: {response.status_code}")
        logger.info(response.text)

        return {
            'statusCode': response.status_code,
            'body': response.text
        }

    except Exception as e:
        print(f"Error: {str(e)}")
        return {
            'statusCode': 500,
            'body': json.dumps({'error': str(e)})
        }
```

Στέλνουμε τους winning clients με τα στοιχεία τους στον Drone provider (REST endpoint που τρέχει σε ένα EC2 instance)

- Fourth lambda function : Στέλνουμε απάντηση στους clients για τα αποτελέσματα του γύρου.

```python
import json
import logging
import boto3

sns_client = boto3.client('sns')

# Enable logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):

    data = {}

    logger.info(event)
    for w in event['winners']:
        data[w['id']] = 'delivered'

    for l in event['losers']:
        data[l['id']] = 'declined'

    response = sns_client.publish(
        TopicArn = 'arn:aws:sns:us-east-1:593367212756:ClientSNS',
        Message = json.dumps(data)
    )

    return event
```

Μαρκάρουμε κάθε client που συμμέτειχε με deliver ή declined ανάλογα με τον αν νίκησε το bid του ή όχι

Και προωθούμε τα αποτελέσματα στο Notification Service που θα ειδοποιήσει όλους τους εγεγραμμένους clients

C2 (Drone Provider's side) :

Δέχεται την λίστα με τους winning clients, κάνει schedule τις παραγγελίες και τυπώνει τα αποτελέσματα όπως φαίνεται

provier.py

```python
from flask import Flask, request
import logging
import json

log_file_path = 'provider_logs.txt'
app = Flask(f"Drone Provider")

#setup logging stuff
formatter = logging.Formatter('Update - %(asctime)s: %(message)s'.format(), datefmt='%Y-%m-%d %H:%M:%S')
handler = logging.FileHandler(log_file_path)
handler.setFormatter(formatter)
app.logger.addHandler(handler)
app.logger.setLevel(logging.INFO)

round_cnt = 1
distance_weight = 0.75
bid_weight = 0.25

def deliver_products(clients):

    #calculate the provider's utility for each client
    for client in clients:
        client['utility'] = round(distance_weight * (10 / client['distance']) + bid_weight * client['bid'], 2)
        print(client)
    ordered_clients = sorted(clients, key = lambda x: x['utility'], reverse = True)
    print(ordered_clients)

    global round_cnt
    app.logger.info(f'*** Schedule in order for round {round_cnt} ***:')
    for client in ordered_clients:
        app.logger.info(f"Client {client['id']} with distance: {client['distance']}km, Bid: {client['bid']} and total utility: {client['utility']}")
    round_cnt += 1

@app.route('/', methods=['POST'])
def receive_orders():
    data = request.get_data().decode('utf-8')
    print(data)
    deliver_products(request.json.get('clients'))
    return 'ok', 200

if __name__ == '__main__':
    app.run(host='0.0.0.0',port=5000, debug=False)
```

Κάνουμε schedule τις παραγγελίες των winning clients
σύμφωνα με την φόρμουλα που βρήκαμε στο θεωρητικό κομμάτι.
Δίνεται μεγαλύτερη βαρύτητα στην απόσταση μέχρι την τοποθεσία του client
και λίγο βαρύτητα στο μέγεθος του bid που πλήρωσε ο client

Αποθηκεύουμε τα αποτελέσματα σε ένα log file για να τα αναλύσουμε αργότερα (αφού ολοκληρωθούν όλα τα rounds)

Ο Drone provider ακούει στο default URL που δημιουργείται όταν ξεκινάει να τρέχει το EC2 instance

C3 (Client's side) :

Κάθε ένας από τους 20 clients είναι ένα process που τρέχει ξεχωριστά σε ένα EC2 instance. Ό κάθε client ακούει στο δικό του port στο default URL του EC2, για να μπορέσει να πάρει απαντήσεις από τον retailer μέσω ενός SNS.

Στην αρχή του πρώτου γύρου :
1. κάθε client αρχικοποιεί τις μεταβλητές του με τυχαίες τιμές :
 bid = απο 5 έως 25, το ποσό που θα υποβάλει αυτό το γύρο
 wait_time = 3 με 7 λεπτά, ο χρόνος που είναι διατεθειμένος κάθε client να περιμένει (μέχρι να κάνει exit->sad)
 distance = 0.5 έως 50 km, η απόσταση του

2. Κάνει subscribe στο SNS topic από το οποίο θα λαμβάνει ειδοποιήσεις από τον retailer και περιμένει να λάβει το subscription confirmation

3. Στέλνει το bid του για αυτό το γύρο στην ουρά για να το πάρει ο retailer.

```python
def init_profile():
    bid = random.randint(5,25)
    wait_time = random.randint(180,420) #wait time in seconds (3-7 mins)
    distance = round(random.uniform(0.5, 50.0), 1)
    init_time = time.time()
    curr_round = 1
    return bid, wait_time, distance, init_time, curr_round

def sns_subscribe():
    response = sns_client.subscribe(
            TopicArn = 'arn:aws:sns:us-east-1:593367212756:ClientSNS',
            Protocol = 'http',
            Endpoint = f'http://ec2-34-235-153-85.compute-1.amazonaws.com:{port}'
    )

def send_request_sqs():
    data = {
        "id": client_id,
        "distance": distance,
        "bid": bid
    }
    response = sqs_client.send_message(
        QueueUrl = 'https://sqs.us-east-1.amazonaws.com/593367212756/BidQueue',
        MessageBody = json.dumps(data)
    )
    global curr_round
    app.logger.info(f'My bid for this round (round {curr_round}) is {bid}')
    curr_round += 1


if __name__ == '__main__':
    #initialize client profile
    bid, wait_time, distance, init_time, curr_round = init_profile()
    app.logger.info(f"Initialized: Distance {distance}km. I'll wait for {wait_time} seconds.")
    sns_subscribe()
    send_request_sqs()
    app.run(host='0.0.0.0', port=port, debug=False)
```

1. Αρχικοποίηση μεταβλητών

2. Εγγραφή στο SNS topic

3. Αποστολή του bid στην ουρά

Όπως είπαμε παραπάνω κάθε client ακούει στο δικό του port και παίρνει απαντήσεις από τον retailer για τα αποτελέσματα του κάθε γύρου.

```python
@app.route('/', methods=['POST'])
def sns_endpoint():
    if request.json.get('Type') == 'SubscriptionConfirmation':
        # Extract the SubscribeURL from the SNS confirmation message
        subscribe_url = request.json.get('SubscribeURL')
        response = requests.get(subscribe_url)
        app.logger.info('Subscribed to SNS topic')
        print("Confirmation response:")
        print(response.text)
    else:
        data = request.get_data().decode('utf-8')
        print(data)
        message = json.loads(request.json.get('Message'))
        if message.get(str(client_id)) == "delivered":
            app.logger.info("Exiting : Product delivered (happy)")
        elif message.get(str(client_id)) == "declined":
            if (time.time() - init_time) > wait_time :
                app.logger.info("Exiting : waiting time surpassed (sad)")
            else :
                app.logger.info("Entering next round")
                bid = random.randint(5,25)
                send_request_sqs()
        else:
            print("Exw teleiwsei koumparo!")

    return 'OK', 200
```

Όταν λάβει ένα μήνυμα από το SNS για το sunbscription confirmation, χτυπάμε το link που στέλνεται από το SNS για να γίνει το validation και να μπορέσουμε να λαμβάνουμε μηνυματα

Αν λάβει απάντηση 'delivered', τότε η παραγγελεία του έχει φτάσει και μπορεί να αποχωρήσει

Διαφορετικά αν λάβει απάντηση declined, το bid που υπέβαλε δεν νίκησε και μπορεί
1. Να αποχωρήσει αν έχει ξεπεραστεί το χρονικο όριο που μπορεί να περιμένει
2. Να συμμετέχει στον επόμενο γύρο υποβάλλοντας καινούριο bid.

C4 (Running experiments: logs & graph analysis) :

Στα πειράματα που τρέχουμε, οι τιμές των Bid, distance, wait_time κλπ παράγονται τυχαία και δεν έχουν κάποιου είδους (AI) λογική από πίσω για να μπορέσουμε να βγάλουμε κάποια ‘βάσιμα’ συμπεράσματα για τον τρόπο που ‘σκέφτονται’ οι clients όσον αφορά τα bids (στρατηγική) που χρησιμοποιούν σε κάθε γύρο.

Αποτελέσματα πειράματος :

- Τα αποτελέσματα από την μεριά του Drone provider (logs) έχουν ως :

provider_logs.txt

```
Update - 2024-01-07 19:23:05: *** Schedule in order for round 1 ***:
Update - 2024-01-07 19:23:05: Client 9 with distance: 16.3km, Bid: 25 and total utility: 4.27
Update - 2024-01-07 19:23:05: Client 4 with distance: 19.2km, Bid: 23 and total utility: 3.89
Update - 2024-01-07 19:23:05: Client 3 with distance: 33.0km, Bid: 24 and total utility: 3.86
Update - 2024-01-07 19:23:05: Client 1 with distance: 33.9km, Bid: 22 and total utility: 3.55
Update - 2024-01-07 19:23:05: Client 15 with distance: 39.7km, Bid: 22 and total utility: 3.51
Update - 2024-01-07 19:23:23: *** Schedule in order for round 2 ***:
Update - 2024-01-07 19:23:23: Client 13 with distance: 12.5km, Bid: 24 and total utility: 4.28
Update - 2024-01-07 19:23:23: Client 14 with distance: 16.7km, Bid: 20 and total utility: 3.51
Update - 2024-01-07 19:23:23: Client 10 with distance: 12.3km, Bid: 15 and total utility: 2.94
Update - 2024-01-07 19:23:23: Client 8 with distance: 11.6km, Bid: 14 and total utility: 2.83
Update - 2024-01-07 19:23:23: Client 0 with distance: 29.6km, Bid: 14 and total utility: 2.39
Update - 2024-01-07 19:23:33: *** Schedule in order for round 3 ***:
Update - 2024-01-07 19:23:33: Client 17 with distance: 9.8km, Bid: 25 and total utility: 4.62
Update - 2024-01-07 19:23:33: Client 5 with distance: 10.0km, Bid: 22 and total utility: 4.15
Update - 2024-01-07 19:23:33: Client 2 with distance: 29.2km, Bid: 20 and total utility: 3.29
Update - 2024-01-07 19:23:33: Client 12 with distance: 13.7km, Bid: 15 and total utility: 2.87
Update - 2024-01-07 19:23:33: Client 11 with distance: 14.7km, Bid: 15 and total utility: 2.83
Update - 2024-01-07 19:23:43: *** Schedule in order for round 4 ***:
Update - 2024-01-07 19:23:43: Client 7 with distance: 10.9km, Bid: 11 and total utility: 2.43
Update - 2024-01-07 19:23:43: Client 6 with distance: 43.0km, Bid: 11 and total utility: 1.85
Update - 2024-01-07 19:23:43: Client 18 with distance: 28.0km, Bid: 7 and total utility: 1.35
Update - 2024-01-07 19:23:44: *** All clients have been served ***
```

- Ενδεικτικά τα logs όλων των client από το initialization μέχρι την ολοκλήρωση των δύο πρώτων γύρων είναι:

```
Client 0 - 2024-01-07 19:19:48: Initialized: Distance 29.6km. I'll wait for 402 seconds.
Client 1 - 2024-01-07 19:19:48: Initialized: Distance 33.9km. I'll wait for 200 seconds.
Client 3 - 2024-01-07 19:19:48: Initialized: Distance 33.0km. I'll wait for 398 seconds.
Client 2 - 2024-01-07 19:19:49: Initialized: Distance 29.2km. I'll wait for 403 seconds.
Client 4 - 2024-01-07 19:19:49: Initialized: Distance 19.2km. I'll wait for 181 seconds.
Client 5 - 2024-01-07 19:19:49: Initialized: Distance 10.0km. I'll wait for 300 seconds.
Client 6 - 2024-01-07 19:19:49: Initialized: Distance 43.0km. I'll wait for 293 seconds.
Client 7 - 2024-01-07 19:19:49: Initialized: Distance 10.9km. I'll wait for 236 seconds.
Client 8 - 2024-01-07 19:19:49: Initialized: Distance 11.6km. I'll wait for 310 seconds.
Client 9 - 2024-01-07 19:19:49: Initialized: Distance 16.3km. I'll wait for 358 seconds.
Client 10 - 2024-01-07 19:19:49: Initialized: Distance 12.3km. I'll wait for 363 seconds.
Client 11 - 2024-01-07 19:19:49: Initialized: Distance 14.7km. I'll wait for 375 seconds.
Client 12 - 2024-01-07 19:19:49: Initialized: Distance 13.7km. I'll wait for 257 seconds.
Client 13 - 2024-01-07 19:19:49: Initialized: Distance 12.5km. I'll wait for 268 seconds.
Client 14 - 2024-01-07 19:19:49: Initialized: Distance 16.7km. I'll wait for 402 seconds.
Client 15 - 2024-01-07 19:19:49: Initialized: Distance 39.7km. I'll wait for 291 seconds.
Client 16 - 2024-01-07 19:19:49: Initialized: Distance 35.1km. I'll wait for 193 seconds.
Client 17 - 2024-01-07 19:19:49: Initialized: Distance 9.8km. I'll wait for 324 seconds.
Client 18 - 2024-01-07 19:19:49: Initialized: Distance 28.0km. I'll wait for 291 seconds.
Client 19 - 2024-01-07 19:19:49: Initialized: Distance 15.3km. I'll wait for 189 seconds.
Client 0 - 2024-01-07 19:19:50: My bid for this round (round 1) is 20
Client 1 - 2024-01-07 19:19:50: My bid for this round (round 1) is 22
Client 2 - 2024-01-07 19:19:51: My bid for this round (round 1) is 9
Client 3 - 2024-01-07 19:19:51: My bid for this round (round 1) is 24
Client 4 - 2024-01-07 19:19:51: My bid for this round (round 1) is 23
Client 5 - 2024-01-07 19:19:51: My bid for this round (round 1) is 15
Client 6 - 2024-01-07 19:19:51: My bid for this round (round 1) is 13
Client 7 - 2024-01-07 19:19:51: My bid for this round (round 1) is 17
Client 8 - 2024-01-07 19:19:51: My bid for this round (round 1) is 6
Client 9 - 2024-01-07 19:19:51: My bid for this round (round 1) is 25
Client 10 - 2024-01-07 19:19:51: My bid for this round (round 1) is 7
Client 11 - 2024-01-07 19:19:51: My bid for this round (round 1) is 8
Client 12 - 2024-01-07 19:19:51: My bid for this round (round 1) is 21
Client 13 - 2024-01-07 19:19:51: My bid for this round (round 1) is 22
Client 14 - 2024-01-07 19:19:51: My bid for this round (round 1) is 10
Client 15 - 2024-01-07 19:19:51: My bid for this round (round 1) is 22
Client 16 - 2024-01-07 19:19:51: My bid for this round (round 1) is 12
Client 16 - 2024-01-07 19:19:51: Subscribed to SNS topic
Client 17 - 2024-01-07 19:19:51: My bid for this round (round 1) is 16
Client 18 - 2024-01-07 19:19:51: My bid for this round (round 1) is 9
Client 19 - 2024-01-07 19:19:51: My bid for this round (round 1) is 5
Client 8 - 2024-01-07 19:19:52: Subscribed to SNS topic
Client 11 - 2024-01-07 19:19:52: Subscribed to SNS topic
Client 14 - 2024-01-07 19:19:52: Subscribed to SNS topic
Client 18 - 2024-01-07 19:19:52: Subscribed to SNS topic
Client 1 - 2024-01-07 19:19:56: Subscribed to SNS topic
Client 3 - 2024-01-07 19:19:56: Subscribed to SNS topic
Client 6 - 2024-01-07 19:19:56: Subscribed to SNS topic
Client 0 - 2024-01-07 19:23:05: Entering next round
Client 1 - 2024-01-07 19:23:05: Exiting : Product delivered (happy)
Client 2 - 2024-01-07 19:23:05: Entering next round
Client 3 - 2024-01-07 19:23:05: Exiting : Product delivered (happy)
Client 4 - 2024-01-07 19:23:05: Exiting : Product delivered (happy)
Client 5 - 2024-01-07 19:23:05: Entering next round
Client 6 - 2024-01-07 19:23:05: Entering next round
Client 7 - 2024-01-07 19:23:05: Entering next round
Client 8 - 2024-01-07 19:23:05: Entering next round
Client 9 - 2024-01-07 19:23:05: Exiting : Product delivered (happy)
Client 10 - 2024-01-07 19:23:05: Entering next round
Client 11 - 2024-01-07 19:23:05: Entering next round
Client 12 - 2024-01-07 19:23:05: Entering next round
Client 13 - 2024-01-07 19:23:05: Entering next round
Client 14 - 2024-01-07 19:23:05: Entering next round
Client 15 - 2024-01-07 19:23:05: Exiting : Product delivered (happy)
Client 16 - 2024-01-07 19:23:05: Exiting : waiting time surpassed (sad)
Client 17 - 2024-01-07 19:23:05: Entering next round
Client 18 - 2024-01-07 19:23:05: Entering next round
Client 19 - 2024-01-07 19:23:05: Exiting : waiting time surpassed (sad)
Client 0 - 2024-01-07 19:23:06: My bid for this round (round 2) is 14
Client 2 - 2024-01-07 19:23:06: My bid for this round (round 2) is 6
Client 5 - 2024-01-07 19:23:06: My bid for this round (round 2) is 11
Client 6 - 2024-01-07 19:23:06: My bid for this round (round 2) is 6
Client 7 - 2024-01-07 19:23:06: My bid for this round (round 2) is 13
Client 8 - 2024-01-07 19:23:06: My bid for this round (round 2) is 14
Client 10 - 2024-01-07 19:23:06: My bid for this round (round 2) is 15
Client 11 - 2024-01-07 19:23:06: My bid for this round (round 2) is 12
Client 12 - 2024-01-07 19:23:06: My bid for this round (round 2) is 9
```

```
Client 12 - 2024-01-07 19:23:06: My bid for this round (round 2) is 9
Client 13 - 2024-01-07 19:23:06: My bid for this round (round 2) is 24
Client 14 - 2024-01-07 19:23:06: My bid for this round (round 2) is 20
Client 17 - 2024-01-07 19:23:06: My bid for this round (round 2) is 9
Client 18 - 2024-01-07 19:23:06: My bid for this round (round 2) is 5
Client 0 - 2024-01-07 19:23:24: Exiting : Product delivered (happy)
Client 2 - 2024-01-07 19:23:24: Entering next round
Client 2 - 2024-01-07 19:23:24: My bid for this round (round 3) is 20
Client 5 - 2024-01-07 19:23:24: Entering next round
Client 5 - 2024-01-07 19:23:24: My bid for this round (round 3) is 22
Client 6 - 2024-01-07 19:23:24: Entering next round
Client 6 - 2024-01-07 19:23:24: My bid for this round (round 3) is 12
Client 7 - 2024-01-07 19:23:24: Entering next round
Client 7 - 2024-01-07 19:23:24: My bid for this round (round 3) is 8
Client 8 - 2024-01-07 19:23:24: Exiting : Product delivered (happy)
Client 10 - 2024-01-07 19:23:24: Exiting : Product delivered (happy)
Client 11 - 2024-01-07 19:23:24: Entering next round
Client 11 - 2024-01-07 19:23:24: My bid for this round (round 3) is 15
Client 12 - 2024-01-07 19:23:24: Entering next round
Client 12 - 2024-01-07 19:23:24: My bid for this round (round 3) is 15
Client 13 - 2024-01-07 19:23:24: Exiting : Product delivered (happy)
Client 14 - 2024-01-07 19:23:24: Exiting : Product delivered (happy)
Client 17 - 2024-01-07 19:23:24: Entering next round
Client 17 - 2024-01-07 19:23:24: My bid for this round (round 3) is 25
Client 18 - 2024-01-07 19:23:24: Entering next round
Client 18 - 2024-01-07 19:23:24: My bid for this round (round 3) is 13
```

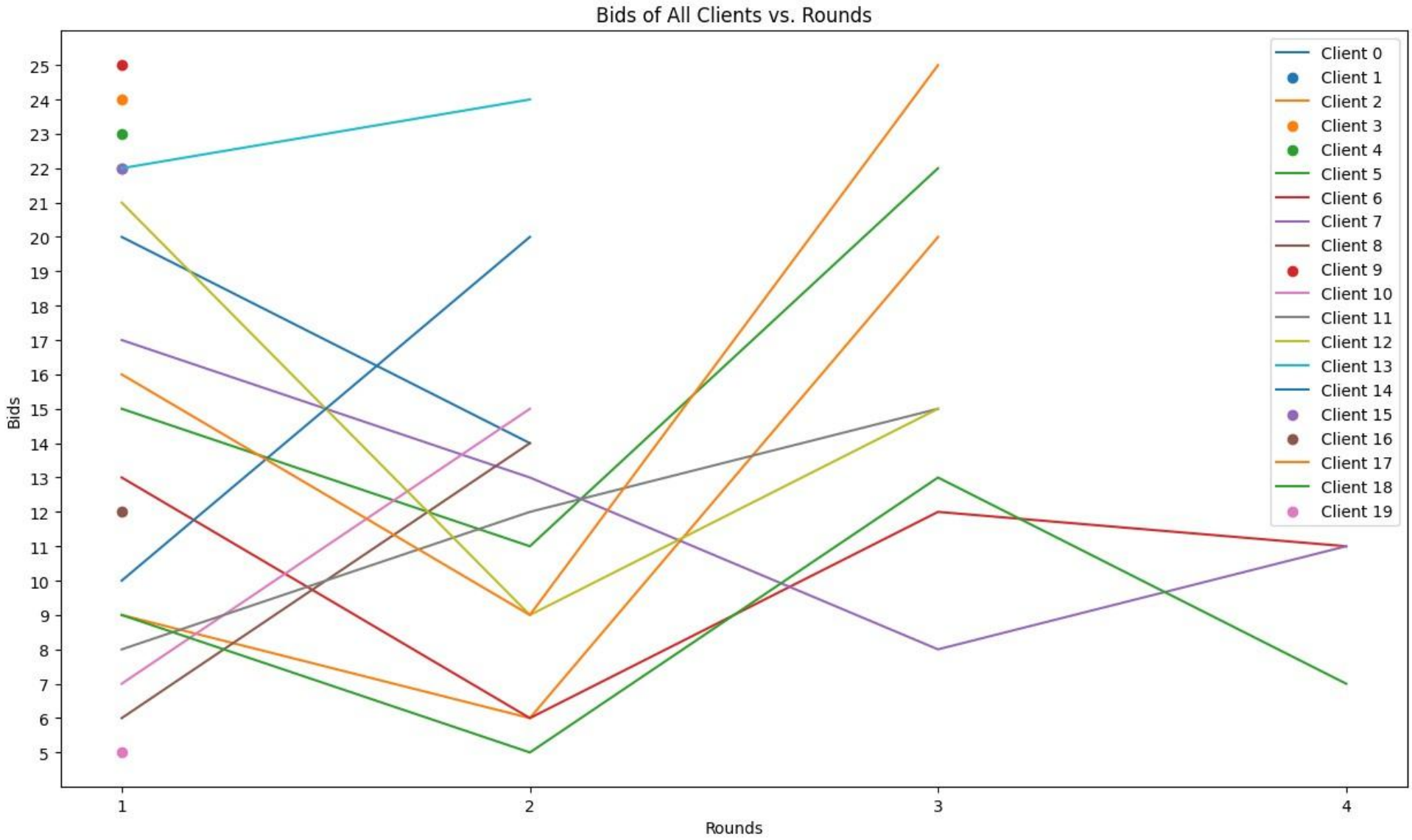
Επίσης ενδεικτικά για τρεις τυχαία επιλεγμένους clients (5,6 και 19) :

```
Client 5 - 2024-01-07 19:19:49: Initialized: Distance 10.0km. I'll wait for 300 seconds.
Client 5 - 2024-01-07 19:19:51: My bid for this round (round 1) is 15
Client 5 - 2024-01-07 19:19:57: Subscribed to SNS topic
Client 5 - 2024-01-07 19:23:05: Entering next round
Client 5 - 2024-01-07 19:23:06: My bid for this round (round 2) is 11
Client 5 - 2024-01-07 19:23:24: Entering next round
Client 5 - 2024-01-07 19:23:24: My bid for this round (round 3) is 22
Client 5 - 2024-01-07 19:23:33: Exiting : Product delivered (happy)
```

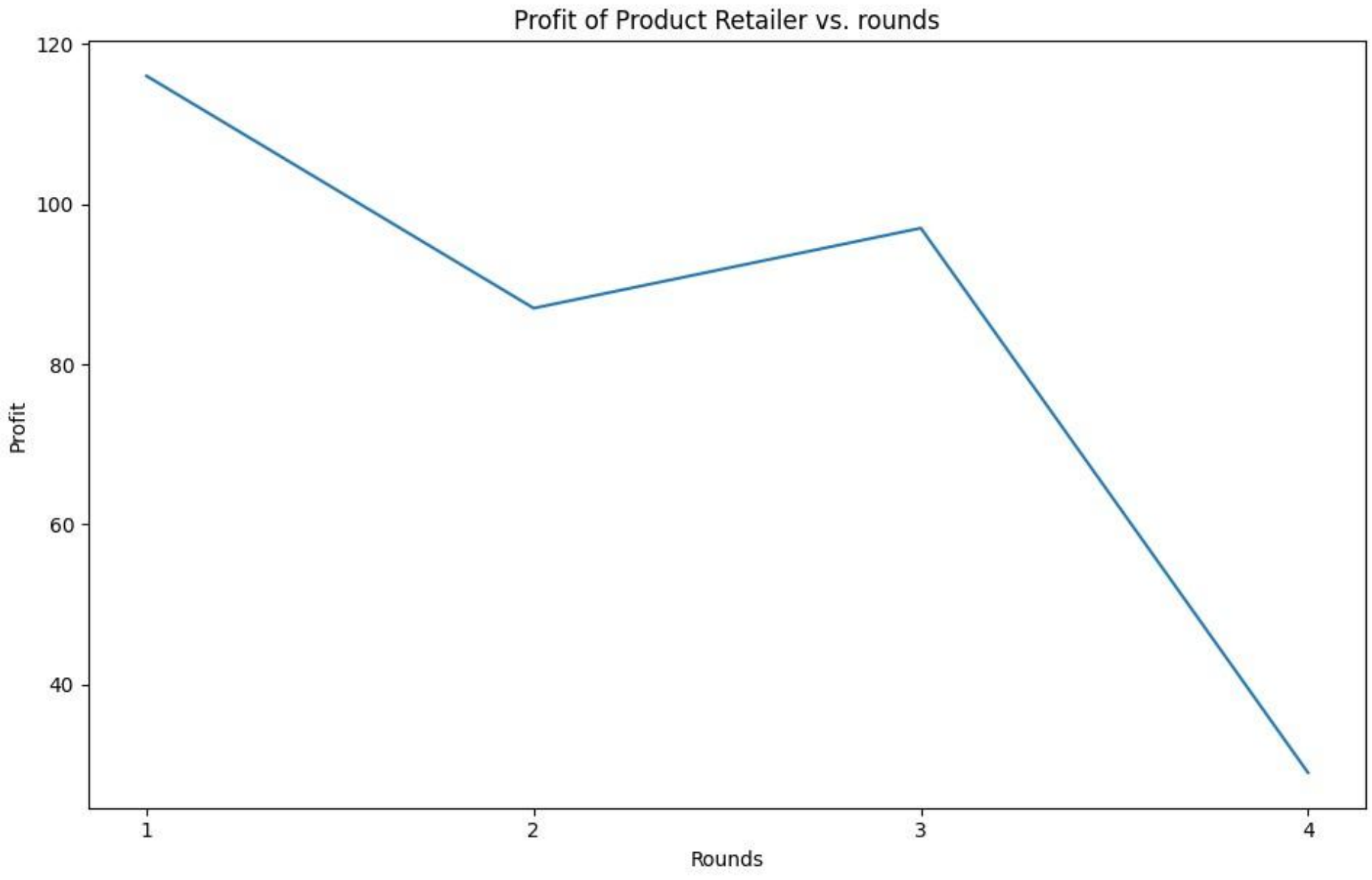
```
Client 6 - 2024-01-07 19:19:49: Initialized: Distance 43.0km. I'll wait for 293 seconds.
Client 6 - 2024-01-07 19:19:51: My bid for this round (round 1) is 13
Client 6 - 2024-01-07 19:19:56: Subscribed to SNS topic
Client 6 - 2024-01-07 19:23:05: Entering next round
Client 6 - 2024-01-07 19:23:06: My bid for this round (round 2) is 6
Client 6 - 2024-01-07 19:23:24: Entering next round
Client 6 - 2024-01-07 19:23:24: My bid for this round (round 3) is 12
Client 6 - 2024-01-07 19:23:33: Entering next round
Client 6 - 2024-01-07 19:23:33: My bid for this round (round 4) is 11
Client 6 - 2024-01-07 19:23:43: Exiting : Product delivered (happy)
```

```
Client 19 - 2024-01-07 19:19:49: Initialized: Distance 15.3km. I'll wait for 189 seconds.
Client 19 - 2024-01-07 19:19:51: My bid for this round (round 1) is 5
Client 19 - 2024-01-07 19:20:22: Subscribed to SNS topic
Client 19 - 2024-01-07 19:23:05: Exiting : waiting time surpassed (sad)
```

- Client Graph:

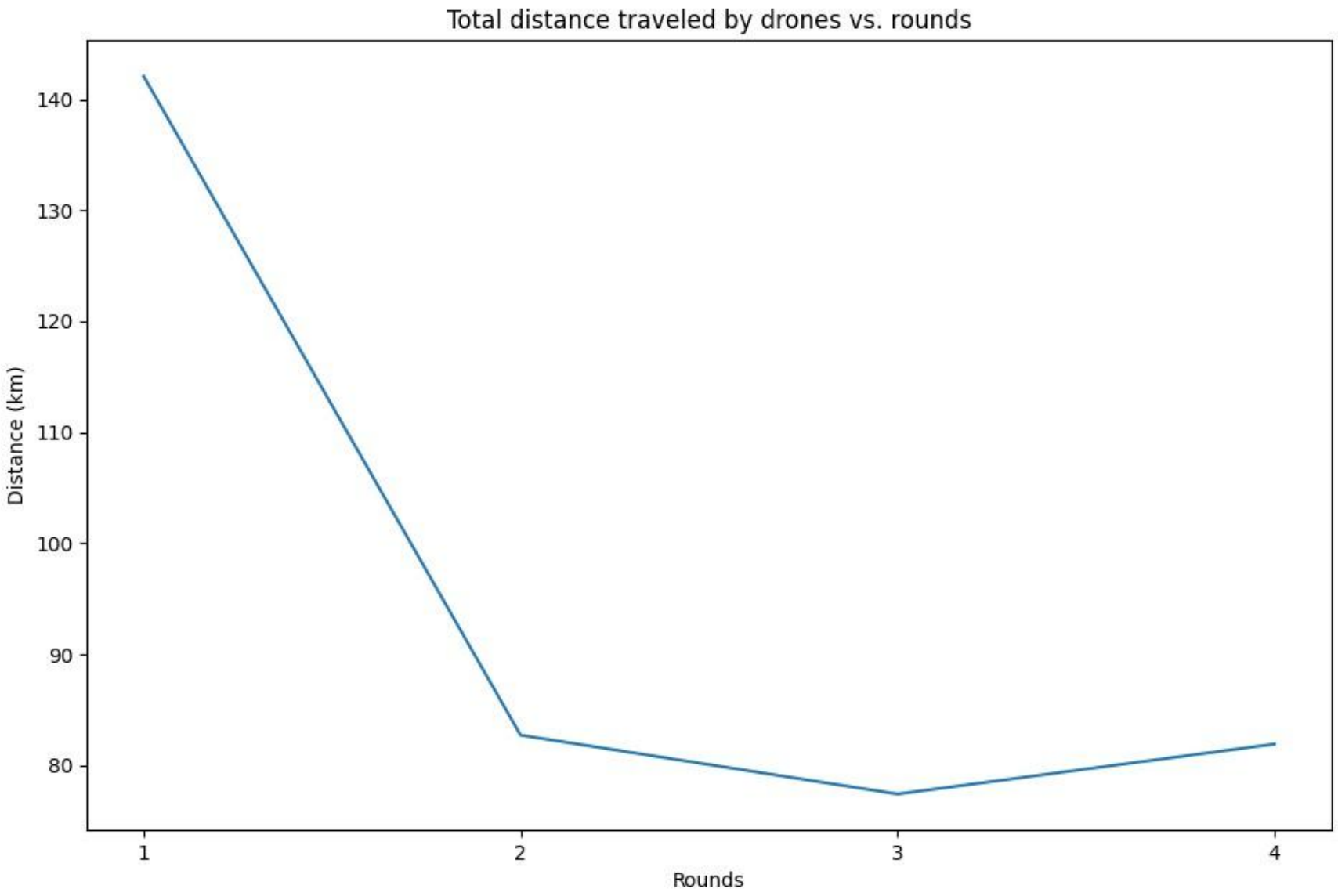


- Retailer Graph:



- Drone provider Graph:

Όσον αφορά το graph για τον Drone provider, δεν αναφέρεται πουθενά στην εκφώνηση της άσκησης, ούτε στην περιγραφή του θεωρητικού κομματιού κάτι για ‘έσοδα’ του Drone provider, παρά μόνο το scheduling των προϊόντων με σκοπό την ελαχιστοποίηση του συνολικού χρόνου παράδοσης.
Έτσι στο σχετικό γράφημα θα αναπαραστήσουμε τα συνολικά χιλιόμετρα που χρειάστηκε να διανύσουν σε κάθε γύρο τα drones για να φτάσουν στην τοποθεσία του κάθε πελάτη. (Χωρίς να συμπεριλάβουμε τον δρόμο της επιστροφής)



Όπως αναφέρει προηγουμένως, οι τιμές των bids παράγονται τυχαία και δεν έχουν κάποια λογική από πίσω.
Όμως αν θεωρήσουμε ότι στο συγκεκριμένο πείραμα οι παίκτες που συμμετείχαν (clients) είναι rational παίκτες και δρουν στρατηγικά, βάση των παραπάνω γραφημάτων και αποτελεσμάτων, παρατηρούμε ότι :

Τα έσοδα του του retailer αλλά και τα συνολικά χιλιόμετρα που διανύουν τα drones από τον πρώτο γύρο στους επόμενους φαίνεται να μειώνονται.
Αυτό θα μπορούσε να σημαίνει ότι μερικοί clients με τοποθεσία μακριά από την αποθήκη του retailer, υποβάλλουν υψηλά bids για να εκμεταλλευτούν το πλεονέκτημα που κερδίσουν στην ουρά προτεραιότητας παράδοσης και έτσι καταλήγουν να κερδίσουν τον πρώτο γύρο.
Από εκεί και έπειτα, στους επόμενους γύρους οι clients που απομένουν αυξομειώνουν στρατηγικά τα bids τους προκειμένου να κερδίσουν τον εκάστοτε γύρο αλλά δεν παρασύρονται στο να κάνουν overbid.