Discussion 2

Routing Optimization System

Problem

Consider that we have an environment having a set of Riders $\boldsymbol{R} = \{\boldsymbol{R_1}, \boldsymbol{R_2}, \boldsymbol{R_3}, \ldots, \boldsymbol{R_N}\}$ and some Restaurants $\boldsymbol{P} = \{\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{P_3}, \ldots, \boldsymbol{P_M}\}$, also known as Pickup locations in our context, and some Orders $\boldsymbol{O} = \{\boldsymbol{O_1}, \boldsymbol{O_2}, \boldsymbol{O_3}, \ldots, \boldsymbol{O_K}\}$, which are being received continuously, one after the other, at the hub. The hub is the central point where all orders are received and the hub has to decide which rider should be assigned what order, such that the overall time to complete all the jobs is minimized. Each order has an impact of $\boldsymbol{t_l}$ minutes on the overall time of a particular rider $\boldsymbol{R_n}$, and an impact of $\boldsymbol{t_g}$ minutes on the overall completion time of all the jobs.

When an Order $\boldsymbol{O_n}$ arrives at the hub for a pickup location $\boldsymbol{P_m}$, the hub filters out the riders that are near to the pickup location $\boldsymbol{P_m}$ and for each rider $\boldsymbol{r_i}$, a fitness value $\boldsymbol{f}$ and the total time $\boldsymbol{rt_i}$, that the rider will take to complete its job after the assignment of the current order, is calculated to find the best possible rider. This process is dynamic i.e. if another order $\boldsymbol{O_{n+x}}$ arrives for the same pickup location, while the earlier one(s) were being processed, the fitness value and the total time to complete the all the jobs, with respect to the new order will also be calculated for all the filtered riders. On the basis of the fitness value of all the riders, the hub will decide which rider is to be assigned what order.

Assumptions

Following are some assumptions for the project.

- The total number of riders is less than the number of orders.
- A rider cannot have more than 5 orders at a time. As for some real businesses, a rider is sent out for deliveries with a maximum of five orders.

Discussed Solution

- The basic structure of the algorithm would be as follows:
    - **Input:** Order, Order_Location, Restaurant_Location, sigma(radius) value, max_orders_allowed
    - **Output:** Rider, Optimized_Route
    - **Steps:**
        1. Niching (Restaurant_Location, sigma, max_orders_allowed): possible_riders
        2. Foreach possible_rider:

        i. Calculate the fitness_function (Rider, Order_Location).
    3. Rank the rider or arange the possible_riders array on the basis of their fitness value.
    4. Select the top most rider.

- **Niching** is the step where we select the riders which could possible minimize the total time taken to deliver a parcel.
- Following are the steps for the neiching process:
    - **Niching (Restaurant_Location, (σ)sigma, max_order_allowed):**
    - **Input:** Restaurant_Location
    - **Output:** Possible_riders
    - **Steps:**
        1. Select all the riders present in the same area.
        2. Initialize an empty array of riders; assign to possible_riders
        3. Foreach rider:
            i. Calculate the distance of the rider from the restaurant location; assign to rider_Distance
            ii. If (ride_Distance <= sigma and rider_order_count < max_order_allowed):
                - Add rider to possible_riders
        4. Return possible_riders.
- **Step 1 (Niching):** This step selects the riders from the overall population and filters out only those fiver riders which could possible minimize the delivery time.
- The above function works by first selecting the riders which are present in the area of the riders. Then, out of those selected riders, it selects the riders present in the radius as specified by **σ** which in our case we have fixed it to 5 km. The value of **σ** can be increased or decreased if we are not getting the enough numbers of riders in a particular area. An empty array of possible_riders is also initialized in the beginning. At a given particular time, we should have at least five riders to compare and find the best possible rider. While filtering out the riders, we also check if the rider is allowed to take more order or not i.e. if the rider has exceeded its max order limit or not which in our case we have decided as 5 i.e. a rider is allowed to have maximum 5 orders at a time. For the time being, we are assuming the values (such as σ, max_orders and the number or riders to search for) as per used by most of the real world companies such as Uber Eats and foodpanda.
- If a rider fits the above criteria, it is added to the possible_riders array.
- The final possible_riders array is returned in the end to be used for the calculation of fitness values which is the step 2 of the basic structure of the algorithm.

- **Step2: Calculating the fitness value for each rider:** This part of the algorithm is not finalized yet in a proper format but the discussed points are mentioned below.

- When an order O_n arrives for a restaurant (Pickup Location)P_M, then after the niching step, the hub will calculate the total time take by each rider got in the niching step, to complete all its jobs in the shortest possible time.
- For the above step the above process will calculate the total route time, after the assignment of the current Order O_n. The algorithm will do this for every rider (Global Search) and every possible network of routes for a rider (Local Search).
- For the above step we can also use Greedy algorithm which tends to find the globally optimum solution while looking for the best solutions locally at each step.
- After the calculation of total route time for every rider, the riders will share their information with each other rider. For the purpose of sharing the information we can use PSO (Particle Swarm Optimization).
- The above process of analyzing the each rider for every incoming order is dynamic i.e. if another order O_{n+x} arrives for the same pickup location, while the earlier one(s) were being processed, the fitness value and the total time to complete the all the jobs, with respect to the new order will also be calculated for all the filtered riders.
- The process explained in the above step and be significantly improved if we are able to predict when and how many new orders are going to arrive for a particular pickup location P_M. For this purpose, we have discussed some options such as exponential distribution, linear regression and Timeseries.
- **Exponential distribution** tells us the time difference between the occurrences of two events. We can predict when the new order is going to come. For exponential distribution, it is a requirement that the events must occur at a constant rate.
- **Time series** can be used to predict how much orders are about to come. Time series can only predict trends and seasonality. While **Regression** can manipulate different features and build a model that would predict the upcoming orders on the basis of the provided features. But, Linear regression can work with only linear data, so any irregularity in the data would produce a significant error in the model.

Question

- Should we work on minimizing **slack time** or **flow time**? Since we are working on a problem similar to food ordering type of businesses which means that there is no completion time of the overall jobs of the system as the process of ordering food is a continuous i.e. orders will keep on coming and will never stop. Whereas a rider, being a single unit, does have a job completion time since it cannot have more than 5 orders at a time which defines the maximum job a rider can perform.
