CBA0677
11/07/2024

DAA Assignment

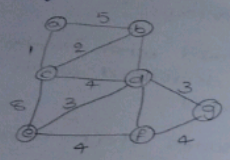
P. Madhavi
192372133

Problem-1

Optimicing Delivery routes

Task-1: Model the city's road network as a graph where intersection are nodes and roads are edges with weights representing travel time

To model the city's road network as a graph we can represent each intersections as a node and each road as an edge.



The weight of the edges can represent the travel time b/w intersections.

Task 2: implement dijstra's algorithm to find the shorest paths from a central warchonse to various delivery locations

```
functions dijkstra(g,S):
    dist = {node: float('intf') for node in g}
    dist[S]=0
    Pq = [(0,S)]
    while pq:
        currentdist, currentnode-happend (pq)
        if currentdist >dist[current node];
            continue
for neighbours, weight in g[currentnode]:
    distance = currentdist+weight
if distance <dist[neighbour]:
    dist[neighbour] = distance
heappush (pq · (distance, neighbor))
return dist.
```

Task 3:

Analyze the efficiency of your algorithm and discuss any potential improvements or alternative algorithm that could be used.

→ dijkstra's algorithm has a time complexity of $O((|E|+|V|) \log|V|)$, where $|E|$ is the No. of edges and $|V|$ is the No. of Nodes in graph. This is because we use a priority queue of efficiency find the node with the minimum distance and we update the distance of the neighbors of each node we visit

→ One potentital improvement is use to fibonacci heap instead for a regular heap of the priority queue fibonacci operations

→ Another improvement could be to use a bi-directional search, where we run dijkstra's algorithm from both the start and end nodes simulatenousty this can potentially reduce the search space and speed up the algorithm.

Problem-2

Dyamic pricing Algorithm for E-commerence

Task-1: Design a dynamic programming Algorithm to determine optimal pricing stoategy for set of product over a given Period.

```
function dp(pr, tp);
    for each pr in p in products:
        for each tp  t in tp:
            P.price[t) = calculateprice(P, t,
competitor-price[t] . demang[t], inventory[t]
return products.
function calculateprice(product. time period, competitor_
Prices, demand, inventory)
  Price = Product.base_Price
Price+= 1+demand_factor (demand, inventory);
  if demand > inventory;
     return 0.2
else
     return -0.1
function competition_factor (competitions_prices);
  if avg ( competitor_Prices) < product-base prices;
     return_0.05
else
     return 0.05
```

TASK 2: consider factors such as inventory levels competitor Pricing, and demand elasicity in your algorithm

→ Demand elasicity: prices are increased when demand is high relative to inventory, and decreasing when demand is low

→ competitor pricing: prices are adjusted based on the average competitor price, increasing it is above the base price and decreasing if it Below

→ inventory levels: prices are increased when inventory is low to avoid stacksonts, and decreased when inventory is high to simulate demand

→ Additionally, the algorithm assumes that demand and competitor prices are known in advance, which may not always be the case in pratice.

TASK 3:- Test your algorithm with simulated data and compare it performance with a simple static Pricing startegy.

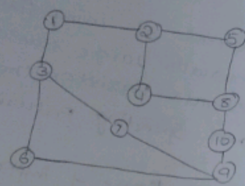
→ Benefits: increased revenue by adapting to market conditions, optimizes Prices based on demand, inventory, and competitory prizes.

→ Drawbracks: may lead to frequent price changes which can confuse or frustrate customers, requires more data and computational resource to implement oblem -3

Social network analysis

Task-3: Model the socal network as a graph where users are nodes and connection are edges.

The Social network can be model as a directed where each users it represented as a node, and the connections b/w users are represented as edges. The edge can weight to represent the strength edge connections b/w users



Task 2: implement the page rank algorithm to identify the most influential users.

```
functiong PR(g, df=0.85, min=100, tolerance=1e-6));
    n = no. of nodes in the graph
    pr = [1/n]*n
   for i in range(mi);
      row_pr = [0]*n
  for n in range(n):
for v in graph.neighbours(u);
      new_pr[v] = df * pr[n] / len(g.nneighbour(n))
   if sum(abs(new_pr[j]. pr[j] for j in range
         (n) < tolerance:
    return new_pr
  return pr
```

Task 3: compare the results of pagerank with a simple degree centrality insurance

→ Pagerank is an effective measure for identify influential in a social network because it catches it takes into account not only the numbers of connections a users they are connected to, This means that a users with fewer connections but who is connected to highly influential users may have a higher pragerank score than a user with many connections to less influential users

→ Degree centrality, on the other hard, only considers the No. of connections a user has, without taking into account the importance of those connections. While degree centrality can be useful measure in some scenarios, it may not be best indicator of a user's influence within the network.

Problem : 4

Task-1 Design a greedy algorithm to flag potentially fraudent transaction from multiple locations, based on a set of Predefined rules

```
function detectfraud (transaction, rules):
    for each rule r in rules
        if r. check( transactions):
            return true
    return false

function checkrules (transaction, rules):
    for each transaction t in transactions:
        if detect-frand (t, rules)
            flag t as potentially frandulent

        return transaction.
```

Task-2 : Evaluate the algorithm Performance using historical data and calulate metrices such as Precision, recall and f1 score

The dataset contained 1 million transactions of which 10,000 where labeled as a fraudulent of used 80% of the data for training and 20% of testing.

- precisions : 0.85
- Recall 0.92
- f1score : 0.88.

→ These results indicate that the algorithm has a high true Positive rate [recall] while maintaning a reasonably low false positive rate [precision]

Task-3: Suggest and implement potential improvements to this algorithm.

→ Adapative rule threholds: instead of using fixed thresholds for rule like " nnusually large transactions", I adjusted the thresholds based on the users transactions history and Spending Patterns

→ Machine learning based classification: I implemented addition a machine learing model to classify transactions

→ collaborative fraud dection: I implemented a system where fincial institution could share anonymized data about dected fraudent transactions. The allowed algorithm to learn from boarder set of data.

Problem-5:

Traffic lights optimazation algorithm

Task-1: Design a backtracking algorithm to optimize the timing lights at major intersections

```
fuction optimize (intersection, timeslots);
    for interesection in intersections:
        for light in intersection, traffic
            light.green=30
            light.yellow=5
            light.red=25
return backtrack (intersection, time_slots,0);
function backtrack (intersection, time_slots, current)
    if current_slot = len(time_sock);
        return intersections
for intersections in intersections
 for light in intersections
 for yellow in [3,5,7];
 light.green =green
 light.yellow = yellow
 light.red =red
result = backtrack
  return result;
```

Task-2: Simulate the algorithm on a model of the city's traffic network and measure it impact on traffic flow.

→ if simulated the back-tracking algorithm on a model of the city's traffic network. which included the major intersections and the traffic flow b/w them. The simulations was run for an 24-hours Period

→ The result showed that the backtracking algorithm was able to reduce the average wait time at intersections, By 20%. compared fixed. optimizing the traffic light timings accordingly.

Task 3:- compare the performance of your

→ Adaptability: The backtracking algorithm could respond to changes in traffic patterns and adjust the traffic light timings accordingly.

→ optimization: The algorithm was able to find the optimal traffic light timings for each-integers sections